2021

Learning Algorithms for Resource Allocation in Wireless Local Area Networks

Yizhou Luo
University of Wollongong

Learning Algorithms for Resource Allocation in Wireless Local Area Networks

A thesis submitted in partial fulfilment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Yizhou Luo

Bachelor of Degree (Telecommunications)

School of Electrical, Computer and Telecommunications Engineering

March 2021

Statement of Originality

I, Yizhou Luo, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institutions.

Signed

Yizhou Luo

March, 2021

Abstract

Nowadays, wireless Access Points (APs) operating in Wireless Local Area Networks (WLAN) or Wi-Fi networks are densely deployed to satisfy user demands for high data rates. In addition, future Wi-Fi networks will be required to support devices, in terms of data communications and energy delivery, that are operating as part of an Internet of Things (IoT) eco-system. However, the increasing energy consumption and interference of Wi-Fi networks have become key concerns to operators and researchers. To address these concerns, a promising approach is to equip APs with Energy Harvesting (EH) capabilities. Another approach is to exploit channel bonding or transmit power control to increase the capacity of a Wi-Fi network. Channel bonding allows an AP to form a wider channel that offers higher data rates. It also helps increase spectrum efficiency. Lastly, transmit power control allows an AP to either increase the received signal strength at users or reduce the interference to neighboring APs.

The aforementioned approaches, however, lead to a number of problems. First, a bonded or wider channel may cause an AP to be more susceptible to interference from neighboring APs because of the use of overlapping channels. Second, as spectrum resource is limited, APs must be assigned an appropriate number of channels corresponding to varying traffic demands over time. Otherwise, they may be under-provisioned, resulting in delays and poor user satisfaction, or over-provisioned,

resulting in low spectrum efficiency. Third, a high transmit power level may lead to higher interference at APs sharing the same channel and also causes an AP to have higher energy expenditure. This is especially important if an AP is powered by a renewable energy source, because such an AP will experience energy outages frequently in the future if it fails to manage the use of energy, resulting in service suspension. On the other hand, a low transmit power level may result in low data rates or energy harvesting rates at IoT devices. Therefore, it is necessary to develop adaptive resource allocation solutions that assign one or more channel(s) and a transmit power level to APs according to their traffic load or channel state. Unfortunately, most resource allocation solutions to date fail to consider traffic variation and random Channel State Information (CSI). Moreover, they assume CSI is fixed or is known by APs in advance.

Henceforth, this thesis aims to develop novel resource allocation algorithms to address the following problems: (i) when do APs use a single or bonded channel? and (ii) when do APs use a high and low transmit power? The developed algorithms should have a learning ability that allows APs to adapt to random traffic load, imperfect CSI, and time-varying energy arrivals. Critically, they should be readily deployable by current APs. Lastly, APs may only have historical and local information, such as the channel gains of users in the last slot and their own data queue length.

This thesis, therefore, investigates learning algorithms that aim to optimize channel assignment and transmit power control in Wi-Fi networks. It first studies the problem of satisfying random user demands in Wi-Fi networks. It formulates this problem as a Markov Decision Process (MDP) where the problem is to use one or more channel(s) that yield the highest user satisfaction in each state. It then proposes a Deep Reinforcement Learning (DRL) solution for APs to learn the best channel bonding policy in order to satisfy their user demand and minimize interference. The solution only relies on historical traffic load and is adaptive to varying traffic load.

Next, this thesis considers charging Radio Frequency (RF)-energy harvesting IoT devices that operate in Wi-Fi networks. These kind of energy users are able to harvest ambient RF-energy whenever a solar-powered AP transmits. They have a certain amount of energy requirement. The aim is to determine the transmit power level of a solar-powered AP in order to satisfy (i) the data rate requirement of legacy data users and (ii) the energy requirement of RF-energy harvesting devices. However, the challenge is that the energy arrival rate of a solar-powered AP and the channel gain to devices are random and time-varying because of fading effect. Current works that consider both types of users do not consider random energy arrivals, and assume an AP has perfect knowledge, e.g., current channel gains to RF-energy harvesting devices. To this end, this thesis models the transmit power allocation problem as an MDP, and solves it using a DRL solution. In addition, this thesis uses Gaussian Process Regression (GPR) to predict network dynamics, e.g., energy arrivals. After that, it uses a Model Predictive Control (MPC) approach to determine the best transmit power for future slots. Both solutions do not require real-time knowledge of the CSI of devices and the energy arrival rate to an AP.

Lastly, this thesis considers approaches to maximize the Energy Efficiency (EE) of an AP subject to random interference from its neighboring APs. The aim is to maximize the EE of an AP while minimizing its data queue delays and overflows. This thesis presents a DRL-based solution to assign one or more channel(s) to an AP and to determine its transmit power level. Unlike previous works, the proposed solution considers random traffic loads and channel gains. Also, it uses only local information available at an AP; i.e., its data queue length. Using an analytical model, this thesis first shows how different traffic loads, channels and transmit power levels impact the EE experienced by an AP. Then, this thesis outlines a model-free MDP. The MDP characterizes the state of an AP, its action, and a reward that is a function of the delay and EE experienced by an AP. It employs a DRL-based solution to derive the best action/policy that yields the maximum reward for each state.

Acknowledgments

I would like to extend my heartfelt gratitude to people who have offered me support and encouragement during my PhD study.

First of all, my genuine appreciation goes to my supervisor A/Prof. Kwan-Wu Chin for his sustained concerns and constructive suggestions. It is his professional guidance and strict requirements that have helped me lay a solid foundation for academic research. Besides, he spared no effort to review my papers and provided prompt feedback even when he is tied up. Without his comprehensive support, I could not have reached this far.

Besides, I gratefully acknowledge all the anonymous reviewers and editors of the journals for their constructive comments, which make my works solid.

Finally, I am indebted to my parents, granny and fiancée for their love and unceasing encouragement during my PhD study. My sincere thanks go to my friends Honglin Ren, Lei Zhang and Yuan Cui, who make my life joyful.

Contents

List of Figures

List of Tables

Abbreviations

WLAN	Wireless Local Area Network
AP	Access Point
BS	Base Station
EE	Energy Efficiency
BSS	Basic Service Set
EH	Energy Harvesting
RF	Radio Frequency
IoT	Internet of Things
CSI	Channel State Information
MDP	Markov Decision Process
DRL	Deep Reinforcement Learning
DQN	Deep Q-network
MPC	Model Predictive Control
GPR	Gaussian Process Regression
CTMC	Continuous Time Markov Chain
DDQN	Dueling Deep Q-Network
CCA	Clear Channel Assessment
HAP	Hybrid Access Point
MBS	Macro Base Station

FBS	Femto Base Station
MINLP	Mixed-Integer Non-Linear Programing
CSMA	Carrier Sense Multiple Access
D2D	Device-to-Device
SINR	Signal to Interference plus Noise Ratio
MILP	Mixed-Integer Non-Linear Programming
RSS	Received Signal Strength
QoE	Quality-of-Experience
CNN	Convolutional Neural Network
MCS	Modulation and Coding Scheme
POMDP	Partially Observable Markov Decision Process
LSTM	Long Short-Term Memory
OBSS	Overlapping BSS
SGD	Stochastic Gradient Descent
TRL	Tabular RL
WCO	Widest Channel Only
PCO	Primary Channel Only
SNR	Signal-to-Noise Ratio
RBF	Radial Basis Function
ADC	Analog-to-Digital Converters
BCO	Bonded Channel Only
R	Random

Chapter 1

Introduction

1.1 Background

IEEE 802.11 Wireless Local Area Networks (WLANs), also known as Wi-Fi networks, have been widely deployed outdoors and indoors, such as offices, shopping malls, and airports, to provide users with high-speed Internet connectivity. In particular, operators are beginning to deploy dense Access Points (APs) or small cell Base Stations (BSs) to satisfy the explosive growth in mobile data traffic [4]. To meet the urgent demands for high data rates, by 2022, a total of 549 million APs will be in operation. A key issue of concern is their energy consumption, where the radio on APs accounts for more than 80% of their energy consumption [5]. This fact motivates research into improving not only the throughput but also the Energy Efficiency (EE) of Wi-Fi networks, where EE is defined as the amount of transmitted data (in bits) using per Joule of consumed energy [6].

Fig. 1.1 shows an example Wi-Fi network. APs have their own coverage. Each AP and its associated users, sensors and devices form a Basic Service Set (BSS); they operate on the same channel and some BSSs may overlap spatially. APs are responsible for transmitting/receiving packets to/from their users or sensors. The central controller manages the Wi-Fi network, where it monitors the traffic load of

APs and assigns them one or more channels.

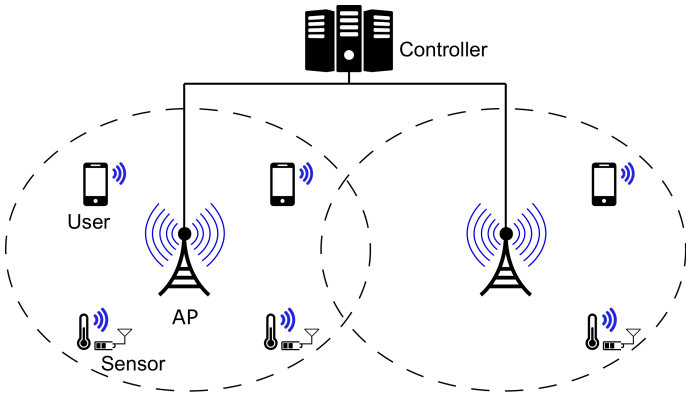

Figure 1.1: An example Wi-Fi network.

Table 1.1 lists key features of current and future Wi-Fi systems. For example, channel bonding allows an AP to combine several channels to form a single logical channel with a wider bandwidth. Hence, APs are able to achieve higher data rates. This is because the capacity of a link is directly proportional to the available bandwidth. A higher data rate leads to smaller transmission delays and higher EE [7] [6]. Transmit power control is another approach to improve the data rate and EE of APs, where an AP can increase its transmit power to improve the received signal strength at users. Alternatively, an AP could reduce its transmit power to minimize its energy consumption or interference to its neighboring APs [6]. Energy efficiency is also becoming a concern [8]. To this end, future WLANs are likely to incorporate Energy Harvesting (EH) APs in order to reduce carbon emissions and operating expenditure [9]. In addition, future Wi-Fi networks can be used to sustain the operation of Radio Frequency (RF)-energy harvesting Internet of Things (IoT) devices; see [10] for a prototype that uses transmissions from an AP to power an on-board camera and temperature sensor. These IoT devices are able to harvest ambient RF-energy whenever the AP transmits to its *legacy* users.

Table 1.1: A summary of key features of current and future Wi-Fi networks.

Features	**Examples**	**Benefits**
Channel bonding	[3, 11]	Increase data rate, energy efficiency; reduce latencies.
Transmit power control	[6]	Increase data rate, energy efficiency, coverage; reduce interference.
Energy Harvesting (EH)	[9]	Reduce carbon emissions and operating expenditure; allow an AP to power itself.
Radio Frequency (RF)-charging	[10, 12]	Support future IoT devices; reduce wiring cost.

1.2 Problem Space and Motivation

The use of channel bonding and transmit power control gives rise to a number of issues. First, although the use of bonded channels affords more bandwidth, it also results in a higher risk of interference due to limited spectrum. Indeed, as shown in [13], naively bonding channels can lead to low data rates.

Second, transmit power control may reduce the received signal strength and data rate of users. This means an AP with heavy traffic load may have insufficient capacity, resulting in data queue *overflows* and high *delays*. On the other hand, a high transmit power creates more interference at nearby APs.

Third, as the traffic load of an AP is time-varying [1], it needs to optimize its transmit power and usage of bonded channels. Doing so helps the AP gain more bandwidth or data rate, accordingly. Otherwise, an AP may have insufficient or over-provisioned capacity when its traffic load changes. This is illustrated in Figure 1.2. Under a fixed channel bonding policy, we see that the capacity of AP-2 is over-provisioned, whereas, AP-1 has insufficient capacity to handle its traffic demand. The optimal policy, see scenario-2, is where AP-2 is allocated less bandwidth than AP-1.

To optimize EE [8], APs may be equipped with the energy harvesting capability that allows them to harvest renewable energy, e.g., solar and winds [9]. Figure 1.3 shows a solar-powered AP that serves not only *legacy* data users/devices, which

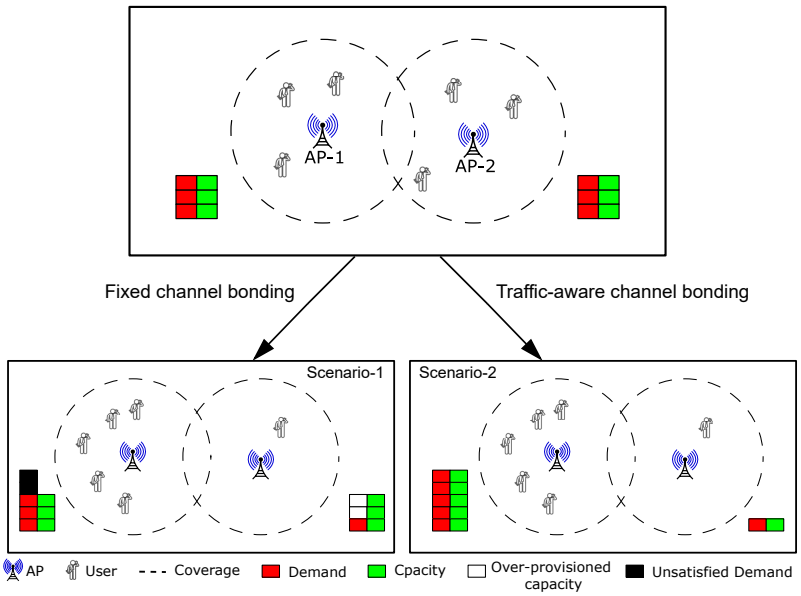

Figure 1.2: An illustration of fixed and traffic-aware channel bonding.

do not have RF-energy harvesting capability, but also nearby IoT devices that are equipped with a temperature sensor and an RF-energy harvester. All nodes operate on the *same* frequency. Whenever the AP delivers data to legacy users, IoT devices harvest RF-energy. The amount of harvested RF-energy is a function of their distance to the AP, time-varying channel gains, how often the AP transmits and also the AP's transmit power; Figure 1.3 shows two possible transmit power levels. A key challenge here is that the transmit power used by the AP is dependent on its available energy, which exhibits spatio-temporal properties. Critically, the AP only has *causal* knowledge of its solar energy arrivals; i.e., the AP only knows its current and past energy arrivals. Another challenge is that IoT devices may be tasked with returning their sample data periodically or require a certain amount of energy to execute their tasks [14]. In both cases, they may not harvest sufficient energy from AP transmissions.

As discussed later in Chapter 2, the main approaches to address the aforementioned issues are integer/non-integer/chance-constraint programming, convex optimization or game theory methods. Most approaches to date do not consider traffic variation. Some naive approaches for channel bonding/transmit power control may optimize a network according to the average or peak traffic demands. In doing so,

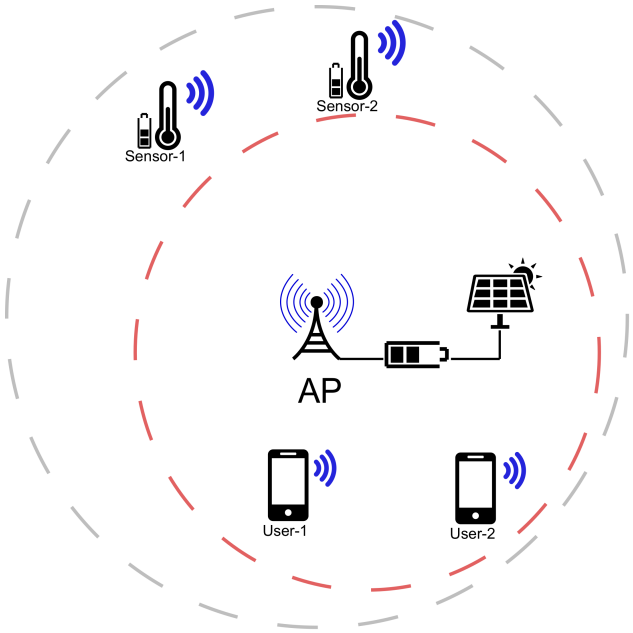

Figure 1.3: An example network with a solar-powered AP and both legacy users and RF-energy harvesting IoT devices. Both sensors receive energy whenever the AP uses a high transmit power level, denoted by the gray circle.

they cannot provide accurate adaptation to traffic changes because the actual traffic may exceed the average value. For adaptive approaches, no one has considered energy efficiency maximization. Critically, they do not consider both RF-energy harvesting and legacy data users in a Wi-Fi network.

Henceforth, this thesis develops machine learning-based radio resources allocation approaches to assign a bonded channel as well as transmit power in order to increase the capacity or energy efficiency of a WLAN. In particular, they will consider the following random quantities: traffic demands, user population, channel gain, energy arrivals.

The key research questions are as follows:

- When and how does an AP bond channel(s)? As mentioned above, channel bonding is a double-edged sword for an AP; it could improve the capacity of the AP, but may also lead to higher interference, which reduces the SINR of users. Hence, it is important to carefully study when and how does an AP bond channel(s).

- How do APs adapt to changing network conditions? One reason for this question is that channel bonding/transmit power policies are dynamic, meaning that the bandwidth, channels and transmit power of APs vary over time. Normally, APs cannot automatically manage their channels and transmit power. Also, without proper training, an AP cannot adapt its policy to traffic. This may result in queue overflows at APs when there is a heavy traffic load, which incurs re-transmissions and high delays. To this end, each AP needs to decide whether it should expand its operating bandwidth, switch to a non-overlapping channel or use a higher transmit power whenever its traffic demand increases.

- How does an AP handle the increased interference when it uses wider channels or high transmit power levels? One approach is to reduce its transmission power. However, this may degrade the SINR of users, which in turn reduces their data rate. Hence, there is a need to jointly optimize channel assignment and transmission power control. More specifically, it needs to consider the following trade-offs: (i) use a bonded channel to obtain a high data rate but incur a high transmit power to overcome interference, or (ii) use a single channel with no interference and minimal transmit power but with a much low data rate. Ideally, an AP should use a bonded channel when there is no interference. However, the main challenge is that the amount of interference is a function of random channel gains and traffic arrivals at neighboring APs.

- How do energy harvesting APs avoid energy shortage and battery overflows when they have *causal* energy arrivals? APs are only aware of current battery level and past information of energy arrivals. This means APs need to learn how to determine the use of energy according to the said information. Otherwise, APs may experience energy shortages when they use a high transmit power or need to transmit frequently. In addition, there will be battery overflows if APs do not transmit frequently.

- How do APs supply energy to nearby RF-energy harvesting devices while not

affecting the data communication of legacy users? RF-energy harvesting devices have sensory tasks, which consumes a certain amount of energy. However, they may not harvest sufficient energy from data communications between an AP and legacy users. Hence, an AP needs to decide when to increase its transmit power level to deliver more energy to RF-energy harvesting devices. However, a critical challenge is that an AP has imperfect Channel State Information (CSI) to devices. Note that it is impractical to collect the actual CSI to IoT devices because they have to be charged first before an AP can send them pilot signals. This incurs significant delays when there are a large number of IoT devices and will impact the throughput of legacy users because all devices and users operate over the same channel. In addition, this process consumes extra energy, which may exacerbate energy shortages at an energy harvesting AP.

1.3 Contributions

This thesis proposes RL-based radio resource allocation algorithms to address the previous problems. In particular, it outlines a number of solutions that allow an AP to learn its transmit power and channel bonding policy over time subject to varying traffic or/and energy arrivals. These solutions are detailed below.

1.3.1 Channel bonding with random traffic demands

This thesis addresses the problem of determining the optimal channel bonding policy that allows APs to adapt to time-varying traffic demands. Specifically, each AP is required to independently learn the number of channels to bond in order to satisfy future traffic demands. A key challenge here is that APs can only rely on historical traffic demands. This problem is of interest because network operators need to manage channel resources according to spatio-temporal user demands [15].

APs equipped with a novel RL-based solution are able to learn the best channel

assignment policy given a varying number of users and interference that stems from using partially overlapping channels. In particular, an AP/agent learns the optimal policy that allows itself to bond channels in order to meet its traffic demands while minimizing interference to other APs. The channel bonding problem is formulated as a Markov Decision Process (MDP) [16], which is then solved by Deep Reinforcement Learning (DRL) or Deep Q-network (DQN) [17].

Past works, see Chapter 2, have used integer/non-linear/chance-constraint programming, convex optimization or game theory methods to bond channels. However, none of them have considered RL approaches [18]. In addition, as the proposed RL approach is run by each AP, APs are able to assign channel(s) by themselves. This means the proposed RL approach is suitable for use in large-scale WLANs. By contrast, most traffic-aware channel bonding algorithms to date allocate channel(s) to APs in a centralized manner. This thesis considers random and unknown traffic demands. By contrast, most prior approaches do not consider traffic variation. Some approaches such as [19] and [20] consider only fixed or known traffic demands. This thesis also considers interference caused by using partially overlapping channels. In contrast, past works assume orthogonal or non-overlapping channels. This consideration is significant because the authors of [21] have shown that assigning non-overlapped channels only may result in poor channel utilization.

1.3.2 RF-charging in Wi-Fi networks

This thesis addresses the problem of satisfying both RF-energy harvesting and legacy data users in Wi-Fi networks. To this end, it presents two solutions to derive a transmit power policy for an EH AP and compare their ability to satisfy both types of users; i.e., either their data rate or energy requirement. The first solution relies on the DQN framework [22], where given an AP's state comprising of its energy level and legacy data user channel gain, it determines the best transmit power that yields the maximum satisfaction for both user types. The second solution uses Model

Predictive Control (MPC) [23], where the AP uses Gaussian Process Regression (GPR) [24], a machine learning method, to predict the AP's future harvested energy and channel gains to legacy users. Lastly, both solutions are readily deployable in current APs.

Past works that consider supporting both types of users, see Chapter 2, have not considered a solar-powered AP, where the proposed solutions ensure efficient usage of an AP's harvested energy. Also, they do not consider imperfect CSI to IoT devices and causal knowledge of energy arrivals. Moreover, their solution has no learning ability and requires non-causal information. On the other hand, this thesis employs machine learning approaches that allow an AP to adapt to its historical CSI and energy arrivals.

1.3.3 Energy Efficient Channel Bonding and Transmit Power Control

This thesis addresses a novel problem of learning the optimal channel bonding and transmit power allocation policy that maximizes the EE of an AP while minimizing its queuing delay and overflows. More specifically, this thesis considers the following trade-offs: (i) use a bonded channel to obtain a high data rate but incur a high transmit power to overcome interference, or (ii) use a single channel with no interference and minimal transmit power but with a much low data rate. Ideally, an AP should use a bonded channel when there is no interference. However, the main challenge is that the amount of interference is a function of random channel gains and traffic arrivals at neighboring APs. In addition, an AP must not maximize EE at the expense of long queues or overflows. To study these trade-offs, this thesis first outlines a novel six-state Continuous Time Markov Chain (CTMC) to gain insights into the channel and transmit power allocation problem over different traffic load scenarios. Next, this thesis models the said problem as an MDP [16] to characterize the state of a Wi-Fi network, and also actions to be undertaken by an AP and

reward for each action. After that this thesis outlines a Dueling Deep Q-Network (DDQN) [25] approach that is run by an AP to solve the formulated MDP.

1.4 Publications

The works of this thesis have been published or submitted at the following venues:

1. **Y.Z Luo**, and K-W Chin. Learning to Bond in WLANs with Random Traffic Demands, *IEEE Transactions on Vehicular Technology*, 69(1), pp.11868-11879, October, 2020.

2. **Y.Z Luo**, and K-W Chin. Learning to Charge RF-Energy Harvesting Devices in WiFi Networks, *IEEE Systems Journal*, 2021. *Accepted, doi: 10.1109/JSYST.2021.3058109.*

3. **Y.Z Luo**, and K-W Chin. An Energy Efficient Channel Bonding and Transmit Power Control Approach for WiFi Networks, *IEEE Transactions on Vehicular Technology. Accepted with minor revisions.*

1.5 Thesis Structure

1. *Chapter 2.* This chapter contains a survey of related works on channel bonding, transmit power control, RF-charging in Wi-Fi networks, and the application of RL in communication systems.

2. *Chapter 3.* This chapter outlines a DRL algorithm that assigns a bonded channel to APs to satisfy their time-varying user demand.

3. *Chapter 4.* This chapter proposes two machine learning-based solutions for a solar-powered Wi-Fi network to satisfy the data rate and energy requirement of users.

4. *Chapter 5.* This chapter presents a DRL solution that assigns transmit power and a bonded channel to an AP in order to maximize the AP's EE while minimizing its queuing delay and overflows.

5. *Chapter 6.* This chapter concludes the thesis, presents the main contributions and possible future research.

Chapter 2

Literature Review

This chapter first presents works on channel bonding; see Section 2.1. After that, Section 2.2.1 discusses works related to wireless charging. Section 2.2.2 reviews works that consider power control/energy management in wireless networks, including WSNs, WPCNs, WLANs and cellular networks. Lastly, Section 2.3 summarizes the application of reinforcement learning in wireless networks.

2.1 Channel Bonding

Works related to channel bonding aim to increase spectrum efficiency, energy efficiency, or/and reduce interference when using bonded channels. The problem is to determine when and how to bond channel(s). As mentioned in Chapter 1 the key issue to consider when bonding channels is the increase in interference among APs, which degrades data rates [13]. In this respect, past works aim to avoid interference by assigning non-overlapping or partially overlapping channels to neighboring nodes. The second issue is bandwidth assignment to APs, which determines their capacity. To be specific, APs need to be assigned with a bandwidth to meet their current traffic demands. This is important because the traffic load of APs is time-varying [1]. A fixed channel bonding policy may result in insufficient or excess capacity. In this regard, prior works on channel bonding can be grouped into two sub-categories: (i)

those that only aim to maximize throughput or minimize interference, and (ii) those that consider traffic load.

2.1.1 Non-Traffic-Aware Channel Bonding

Prior works on channel bonding aim to maximize AP throughput. They assume that APs are always busy and do not consider traffic load variation. Reference [26] and [27] study the problem of when and how a node bonds channels in order to maximize its throughput. Both works employ a continuous-time Markov Chain to model channel availability and then determine which channel should be bonded. In reference [26], the authors use nonlinear integer programming and the branch-and-bound method to determine a channel bonding solution with the minimum interference. Reference [27] studies four channel bonding rules: (i) no bonding and only use an idle channel, (ii) always bond the whole band, (iii) always bond the widest continuous idle channels, and (iv) bond a random number of continuous idle channels. Then, it proposes an algorithm that decides which rule is adopted under different deployment densities.

The work in [28] considers Clear Channel Assessment (CCA) adjustment when an AP bonds channel. APs are pre-assigned a group of partially overlapping channels. The aim is to minimize collisions as well as the number of backoffs. The algorithm calculates an appropriate CCA threshold for each AP according to its link quality and channel occupancy time. The link quality corresponds to the average received signal/interference strength observed by clients. To ensure nodes access the channel fairly, a node will analyze the channel occupancy time of different clients to fine-tune their CCA threshold. After that, each AP independently adjusts its bandwidth under different deployment densities to avoid collisions.

In references [29] and [30], the authors employ game theory to maximize the throughput of APs. The problem is formulated as a game, where each AP is a player, and it adopts different channel bonding strategies. By observing the strat-

egy of neighboring APs, an AP decides the optimal channel bonding strategy that maximizes its utility function. In reference [29], the utility function is defined as a function of the throughput of an AP and its neighboring APs as well as their SINR values. In [29], the utility function is related to the throughput of an AP and the amount of overlap with the frequency chosen by neighbor APs.

Another work in [31] studies the energy consumption and throughput of APs when using different bandwidths. The proposed bandwidth allocation solution aims to maximize the throughput or minimize the energy consumption of an AP when it is transmitting a given file to a user.

2.1.2 Traffic-Aware/Adaptive Channel Bonding

There are also works that aim to design traffic-aware channel bonding algorithms. The main research question is how to determine the bandwidth or capacity of APs given their traffic. The aim is to maximize the probability that APs satisfy their traffic demands. Existing works have also considered other factors such as fairness, balance in capacity, interference or spectrum efficiency.

A recent work [32] proposes a demand-sensitive channel bonding algorithm for IEEE 802.11ac networks. APs are assigned a group of non-overlapping and contiguous channels by a central controller. The aim is to determine a channel assignment/bonding policy that minimizes the usage of bandwidth while meeting the demand of APs. The authors solve the problem in two stages. First, they assume demands are deterministic and then determine the central frequency and bandwidth for APs. The authors use an integer linear program to determine the central frequency of APs. In the second stage, the authors consider random demands. The authors employ chance-constrained programming to minimize the difference between the demand and capacity of APs. In reference [15], APs have time-varying traffic load and periodically exchange their traffic load information with neighboring APs. Using the exchanged information, APs aim to (i) prevent starvation, and (ii) miti-

Table 2.1: Works on channel bonding.

Ref	Channel	Traffic-aware	Tool	Key considerations
[26]	Partially overlapping	No	Integer nonlinear programming, Branch-and-Bound Method	Interference between nodes
[27]	Non-overlapping		Heuristic method	Interference between nodes, deployment density
[28]	Partially overlapping		Clear Channel Assessment (CCA) adjustment	Fairness, collisions
[29]	Partially overlapping		Game theory	SINR of nodes
[30]	Partially overlapping		Game theory	Channel overlaps
[32]	Non-overlapping	Yes	Integer linear/chance-constrained programming	Channel bandwidth, traffic demands of nodes
[19]	Partially-overlapping		Heuristic method, graph theory	traffic demand, SINR or users, channel overlaps, fairness
[15]	Non-overlapping		Heuristic method, graph theory	traffic demand, channel overlaps, fairness
[33]	Non-overlapping		Integer linear programming, backtracking search algorithm, graph theory	Spectrum utilization, traffic demands, fairness
[34]	Non-overlapping		Max-min fairness, graph theory, backtracking search algorithm	Traffic demands, fairness
[35]	Non-overlapping		Deep reinforcement learning	Traffic demands, queue length, latencies

gate interference between neighboring APs.

Reference [33] considers two objectives: (i) to maximize the spectrum utilization of APs, and (ii) to ensure that each AP fairly has the capacity corresponding to its traffic load. The authors apply integer linear programming to assign non-overlapping channels to APs. Different from [32] and [33], reference [34] considers assigning overlapping channels to interfering APs according to the number of associated users. The objective is to (i) ensure the max-min fairness of bandwidth obtained by each link between an AP and its user, and (ii) mitigate interference between links. In a similar work [19], when determining the bandwidth and central frequency of a link, the authors consider a link's traffic demand, SINR and channel overlaps. In particular, the authors schedule links and assign channels according to the estimated interference and throughput of links.

Another game-theoretical work in [20] aims to achieve Nash Equilibrium in which APs have the minimum difference between their capacity and their traffic load. Recently, reference [35] adopts a DRL approach to assign a bonded channel to AP with a random traffic load. The DRL solution aims to assign non-overlapping channels to APs in order to minimize their queue length or latencies.

2.2 Wireless Charging and Transmit Power Control

This section summarizes the works that have considered wireless charging in Wi-Fi networks. Next, this section reviews works on power control in wireless networks.

2.2.1 Wireless Charging in Wi-Fi networks

To date, numerous past works have employed wireless charging, also known as RF-charging technologies, see [36] for details. RF-charging technologies enable a Hybrid Access Point (HAP) to replenish nodes wirelessly. Using their harvested energy,

these nodes sense their environment, which generates data to be transmitted to the sink node or a HAP. In future Wi-Fi systems, an AP is likely to operate as an energy source or HAP for RF-energy harvesting nodes/users; see [10] for a prototype that uses transmissions from an AP to power on-board cameras or temperature sensors. Specifically, the authors consider a traditional AP that uses transmit power of 23 dBm on three antennas with 4.04 dBi gain to power low-power temperature and camera sensors within 20 and 17 feet, respectively. These so-called nodes/users/sensors are able to harvest ambient RF-energy whenever APs transmit data. However, as mentioned in [10], a key challenge is that RF-energy harvesting devices may not harvest sufficient energy from AP transmissions. Hence, a key problem is determining how APs use the wireless channel to ensure RF-energy harvesting devices have sufficient energy to achieve their sensing tasks, while not affecting data communications. For example, an AP needs to determine its transmit power or when to send a power packet for RF-energy harvesting devices. In this regard, this section summarizes works that consider the coexistence of RF-energy harvesting devices and legacy data users, or the coexistence of data communications and energy transfers. Note, these two types of users have different properties. First, RF-energy harvesting devices comprise an energy harvester and battery. They can harvest energy when an AP transmits to its users. Alternatively, an AP needs to send them dedicated power packages. By contrast, data users only have a data rate requirement. To date, only a small number of works have considered supporting these two types of users; examples include [10, 12, 37–40].

In [10], the authors use IEEE 802.11g APs to power battery-free temperature or camera sensors. These sensors have a certain energy requirement. To guarantee that sensors always have sufficient energy, APs inject a power packet whenever their data queue length is shorter than a given threshold. The problem is to determine this threshold for different distances between sensors and an AP.

In [38], the authors consider an OFDMA-based two-tier system with a Macro Base Station (MBS) and a Femto Base Station (FBS). RF-nodes are assigned sub-

carriers dedicated for charging. The authors consider the following tradeoff: the interference caused by an MBS and FBS is beneficial for RF-nodes, but it is harmful to legacy data users. This is because data users will experience a poor SINR, while RF-energy harvesting devices could harvest more energy. The objective of [38] is to use the least amount of energy to (i) deliver sufficient energy to RF-nodes, and (ii) meet the data rate requirement of legacy data users. The challenge is that the channel gain of sub-carriers is different. The authors employ Mixed-Integer Non-Linear Programming (MINLP) to determine sub-carrier and transmit power for both RF-nodes and legacy data users.

Reference [37] considers using several APs to power sensor nodes. APs maintain a packet queue. The objective is to schedule the transmissions of APs in order to (i) maximize the amount of energy delivered to RF sensors, and (ii) minimize the queue length of APs. The challenge is that the packet arrivals and channel gains are random. The authors model the scheduling problem as an MDP. The state is the queue length of APs and the channel gain to sensors. The action is to determine an order for AP transmissions. The reward received by an agent is a function of the sum rate of data users and the amount of energy delivered to sensors.

Unlike [38] [37] [10], reference [12] considers uplinks. Specifically, sensor nodes are able to harvest ambient RF-energy whenever (i) an AP transmits beacons to sensors, and (ii) a neighbor sensor node transmits its data to an AP. Sensor nodes have a constant energy requirement per slot. The objective is to maximize the total network throughput while meeting the energy requirement of sensor nodes in each slot. The problem is to determine the frequency of sending a beacon and the charging time for sensor nodes.

Reference [39] considers only uplink communications. In each slot, a HAP first charges RF-energy harvesting devices and then receives data from both legacy data user and RF-energy harvesting devices. The aim is to (i) maximize the total throughput of the system, or (ii) maximize the minimum throughput of users. The problem is to determine the duration allocated for charging RF-energy harvesting devices

and for receiving data from users.

Reference [40] considers a multiuser MIMO system with several data users and one RF-energy harvesting user. The problem is to determine the maximum number of active beams at a base station for data users under a constant total transmission power constraint. The objective is to (i) maximize the sum-rate of data users, and (ii) ensure that a coexisting RF-energy harvesting user harvests a given amount of energy from beams.

2.2.2 Transmit Power Control

Apart from channel bonding, transmit power control is an alternative approach to improve the EE or throughput of a wireless network. It allows a BS/AP to adjust its transmit power level in order to (i) reduce its energy consumption or interference to other APs, or (ii) to increase the receive signal strength and data rate of users. However, an AP using transmit power control has to consider the following trade-offs: (i) use a high transmit power to obtain a high data rate but incur high energy consumption and interference to neighboring APs, or (ii) use a low transmit power to minimize the interference but with a much lower data rate. A key challenge is that the condition of a channel, the traffic and channel accesses of APs are causal and random.

This section reviews prior works that have considered the said power allocation problem. Examples includes [41–44]. They apply transmit power control to improve the data rate or energy efficiency of a wireless network. Specifically, reference [41] considers transmit power control for an AP when it has a Markovian channel. The channel condition of an AP is modeled as a Markov chain with two states; namely, 'bad' and 'good'. The aim is to minimize the energy consumption of an AP for a given packet error rate of a single mobile user.

In reference [42], the authors consider the impact of Carrier Sense Multiple Access (CSMA) when assigning transmit power to APs. The aim is to minimize the

interference among two neighboring APs, and thereby, increase their throughput and energy efficiency. Users evaluate the Received Signal Strength Indicator (RSSI) from an associated AP and next strongest AP(s), then return this information to its AP via TCP frames. Then, APs exchange information with each other via a controller. Last, with this information, APs adjust their transmit power cooperatively to mitigate the interference to these edge users. The work in [44] considers a WLAN coexisting with a Device-to-Device (D2D) link; they share the same spectrum. The objective is to determine the transmit power of an AP in order to maximize the EE of the AP while avoiding the jamming of the D2D communication.

Many works jointly apply transmit power control and methods such as channel/sub-carrier assignment [43, 45–48], modulation order selection [44, 45, 49], CCA threshold adjustment [50], developing new MAC protocols [42, 51, 52], link schedule [53] and AP sleep control [47]. Among these references, the most relevant references include [43, 46–48], which jointly consider channel assignment and transmit power control in wireless networks. They aim to maximize the energy efficiency or sum-rate of APs/BSs.

Reference [48] assigns sub-carriers to users and then determines the transmit power of sub-carriers in order to maximize the EE of a cell. The work in [46] considers packet routing, transmit power control and channel assignment for meshed APs. The objective is to minimize the energy consumption for transmitting one packet. Reference [43] proposes a joint channel allocation and transmit power control algorithm for densely deployed APs to minimize the interference between them. The optimal transmit power and channel is determined as per the Signal to Interference plus Noise Ratio (SINR) experienced by users. In reference [47], the authors apply Mixed-Integer Non-Linear Programming (MILP) to jointly determine user-AP association, AP sleep mode, transmit power and channel assignment in order to minimize the energy consumption of an AP.

2.3 Reinforcement Learning

Reinforcement Learning (RL) [18] is a branch of machine learning, which is widely applied in many areas such as intelligent control, analysis and prediction. It allows an intelligent agent to gain the knowledge of an environment and makes a decision that optimizes a given objective. RL can also include an artificial neural network to form so-called Deep Reinforcement Learning (DRL) [54]. To date, RL has been applied widely in communication systems; e.g., [54]. In general, an intelligent agent can run at a base station, a transmitter or a central controller. It observes the state of an environment, takes an action and then receives a reward. Here, the environment is represented by a tuple of numeric values which indicates the current or past network status observed by an agent; e.g., channel gain, SINR or Received Signal Strength (RSS). The action taken by an agent usually corresponds to adjusting some network parameters; e.g., power level or channel. The reward received by an agent can be the throughput or capacity of a base station.

This section discusses the application of RL to achieve objectives such as interference mitigation and collision avoidance. Approaches include channel allocation, transmit power control and media access control. Table 2.2 presents a summary of optimization problems in different communication systems that can be solved by employing an RL framework.

2.3.1 Cognitive Radio

Cognitive radio systems offer a natural ground for the application of RL. To utilize radio resources in a more efficient manner, users are classified as primary and secondary users. Primary users are licensed to access a frequency band, while secondary users are able to transmit on the same frequency band if they do not interfere with any primary user. To this end, the major aim of secondary users is to avoid interfering with primary users.

References [55] and [56] aim to minimize the interference between primary users

Table 2.2: Works that apply RL.

Ref	System	Aim	Approach	Method/Tool	Agent	State	Action	Reward
[55]	Cognitive radio	Maximize secondary users' access /collision avoidance	Channel access control	Q-learning	A secondary user	Channel, secondary users	Access a channel or remain idle	Whether there is a collision
[56]		Maximize secondary users' SINR /guarantee primary users' QoS	Transmit power control	DRL	A secondary user	SINR of primary/secondary users	Configure secondary users' transmit power	SINR of primary/secondary users, a threshold value
[57]		Guarantee primary users' SINR/maximizing the QoE of secondary users	Transmit power control	DRL	A secondary user	SINR threshold, interference to other users	Set a SINR threshold for secondary users	Error rates, SINR at receivers, packet loss
[58]	Femtocell	Inter-tier interference mitigation/ throughput maximization	Subcarrier assignment	Q-learning	A femtocell base station	Number of neighbour cells, whether interfere with macrocell users or not	Assign subcarriers to users	SINR of links
[59]		Inter-tier interference mitigation/ throughput maximization	Transmit power control	Q-learning	A femtocell base station	Whether interfere with macrocell users or not, transmit power	Configure the transmit power of base stations	SINR value of the macrocell users
[60]		Interference mitigation/ throughput maximization	Subcarriers assignment/ transmit power control	Cooperative Q-learning	A femtocell base station	Interference strength, power usage, number of users on a subcarrier	Assign users to subcarriers and configure the transmit power of base stations	Throughput of a femtocell
[61]	Cellular	Maximize the number of available channel	Dynamic channel assignment	DRL	A cellular base station	Channel resource usage	Assign channel to an incoming call	Channel reusing
[62]		Mitigate Inter-cell interference	Downlinks Transmit power control	DRL	A cellular base station	Transmit power of base stations, average receive power strength, interference strength on users	Configure transmit power	Throughput of cell or users
[63]		Mitigate Inter-cell interference	Users' uplinks power control	DRL	A central controller	SINR and interference strength on users	Configure transmit power to users	Throughput of users
[64]	Satellite	Optimize link parameters	Optimize modulation order, encoding rate and transmit power	DRL	Two neural network based modules on controller	Channel conditions	Tune the parameters shown in column 4	Predicted by neural networks based on different objectives
[65]		Minimize the number of blocked requests	Dynamic channel assignment	DRL+CNN	A controller	Number of activated users, channel resource usage, new request	Assign a channel to a new request or decline the request	Whether a new request is blocked
[66]	WLAN	Mitigate interference/maximize throughput	Channel assignment/transmit power control	Q-learning	An AP	SINR value of a user	Assign channel and configure the transmit power of APs	Throughput of a BSS
[67]		Mitigate interference/maximize throughput	Transmit power control	Actor-Critic	A transmitter	Number of packets in a queue, the SINR of receiver	Configure transmit power	Packet overflow and transmission failure
[68]		Mitigate interference/maximize throughput	Transmit power control	DRL	An AP	Current power level, interference to/from neighbors	Configure transmit power	Throughput of users/the capacity loss at neighbour due to interference
[69]		Minimize error rate/Maximize throughput	Optimize channel bandwidth, guard interval size and frame aggregation level.	SARSA	An AP	The value of the said parameters	Configure link parameters	Bit error rate and frame error rate.
[70]		Maximize successful transmissions	Dynamic spectrum access	DRL	A transmitter	Status of past transmissions, channels used and channel capacity	Access a channel or remain idle	Transmissions
[71]		Maximize successful transmissions	Dynamic spectrum access	DRL	A transmitter or receiver	Channel condition statistics	Access/sense a channel	Transmissions

and secondary users. Their idea is to observe the channel access behavior of primary users. For example, these users could observe which channels to access and their transmit power in order to learn how to take an action to avoid collision with primary users. Reference [55] considers channel assignment for secondary users, while [56] determines the transmit power level of secondary users. Both works employ Q-leaning, but reference [56] uses a neural network, which allows an agent to deal with high dimensional state-action space. Secondary users run an agent. The state, action and reward definition of reference [55] [56] is shown in Table 2.2.

Reference [57] proposes a DQN based power control algorithm for secondary base stations. The authors consider the Quality-of-Experience (QoE) of users, which is evaluated by the actual throughput, error rate, packet loss probability and SNR of a video traffic link. Both primary and secondary users have a required SINR threshold. The aim is to guarantee the SINR requirement of users while maximizing the QoE of secondary users. An agent decides the SINR threshold for each secondary user. Then, each secondary user is able to calculate the optimal transmit power according to the SINR threshold of users. The state observed by an agent corresponds to the current SINR thresholds of users and the interference causes to the primary user and secondary users. The action taken by an agent is to determine the SINR threshold of a secondary user. The reward received by an agent corresponds to the QoE gained by secondary users.

2.3.2 Femtocell

RL is also appropriate for femtocell systems. A femtocell is a small, low-power base station that is used to extend the service coverage in indoor environments or at the edge of a cell. Channel resources in femtocell networks are divided into resource blocks containing several subcarriers. A femtocell system always coexists with other telecommunication systems; e.g., a macrocell system. To this end, one key challenge is how to mitigate inter-tier interference between macrocells and femtocells.

To mitigate inter-tier interference in femtocell systems, prior works have considered sub-carriers assignment [58], power control [59] and those that consider both subcarrier allocation and power control [60]. In [58] and [59], the approaches are decentralized, meaning each femtocell is an independent agent and does not exchange any information. Agents learn how to assign subcarriers to their users or configure their transmit power on different subcarriers automatically to mitigate interference. Both works employ Q-learning. The state, action and reward definition of [58] and [59] are compared in Table 2.2.

In reference [60], the authors utilize both subcarriers assignment and power control to mitigate interference. The novelty of [60] is that femtocells are not independent agents as they may exchange some information. Agents maintain a global Q-table to achieve cooperative learning. A key challenge in cooperative learning is that agents may lose generalization in choosing actions as they share the same Q-table. To deal with this challenge, an agent tends to choose subcarriers with a higher channel gain. As the channel gain changes, different agents will not always choose the same action.

2.3.3 Cellular

In a cellular system, a channel only belongs to a user during its service time. The channel will be released after the service terminates. If there is no channel for a new request, then it will be blocked. Another problem is that due to frequency reuse, users in a cell may experience inter-cell interference. These problems require specific channel resource management and transmit power control. The following works employ RL to carry out channel assignment and power control in cellular systems.

Maximizing the number of channels in a cell leads to fewer blocked requests. In this respect, reference [61] uses Q-learning to achieve the said aim. Base stations run an agent. The state observed by an agent is defined by the number of channels

occupied by interfering cells. The action taken by an agent corresponds to assigning an available channel to an incoming request. The reward received by an agent corresponds to how many cells are using an assigned channel. To deal with large space and continuous input parameters, the authors use a neural network to approximate state-action values.

As for interference mitigation, the authors in reference [62] and [63] consider power control of downlinks and uplinks, respectively. Specifically, in reference [62], each base station runs an agent and applies the transmit power control policy of the agent. However, in [63], a central controller that runs an agent adjusts the transmit power of users. Both works use SINR to measure interference. The SINR of a user is associated with its time-varying channel gain, large-scale fading and transmit power. The state observed by an agent corresponds to the transmit power of a base station, the received power, interference strength or SINR of users. The action taken by an agent is to assign transmit power level to a base station. The reward received by an agent is the throughput of cells or users. The novelty of [63] is that the reward is shared by neighboring agents, which encourages agents to cooperate rather than contend for resources. Both works use neural networks to deal with high-dimensional and continuous state-action space.

2.3.4 Satellites

RL is also employed in satellite communications. The simplest satellite system consists of a satellite in orbit and a ground base station. Here, the key challenge is to tackle changing channel conditions due to atmospheric conditions and spacecraft orbital dynamics. Further, the uncertainty in channel conditions produces high dimensional and continuous state-action space, leading to the problems of poor convergence and complex computation.

To address random channel conditions, reference [64] designed a novel RL framework to optimize satellite link parameters such as modulation order, encoding rate

and transmit power. An agent takes an action to tune these parameters. The novelty of this work is that it uses two different neural network blocks to cooperatively choose an action; namely, Block-1 and Block-2. Each block contains several neural networks. The state observed by an agent corresponds to the SNR level at the ground station. Block-1 predicts the value of actions for a certain state, whereas, Block-2 chooses an action that is able to satisfy the maximum value estimated by Block-1. The proposed framework can be used to optimize multiple network parameters in parallel; e.g., bit error rate, throughput, bandwidth usage, spectral efficiency, consumed power, and power efficiency.

Another work provides a solution to deal with high dimensional state-action space. The solution uses a DQN and a Convolutional Neural Network (CNN) [65]. The authors consider a multi-beam satellite system. The aim is to minimize the number of blocked requests that arrive at a satellite. The state observed by an agent is defined by three indicators which are the group of active users, the channels they occupy, and the user that initiated a new request. The action taken by an agent is to assign a channel to a new request or decline the request. The reward gained by an agent depends on whether a new request is blocked. Interestingly, the authors adopt a CNN to extract the useful features from the geographical footprint of beams in order to reduce the dimension of state space.

2.3.5 Wireless Local Area Networks

In Wireless Local Area Networks (WLANs), RL can be used to solve media access control, transmit power control, and channel assignment problems. Each AP has its own coverage and forms a BSS. Users or sensors are randomly located around APs. Some BSSs may overlap spatially. As mentioned in Chapter 1, the problem is to carefully handle the increase in interference between overlapped BSSs, which degrades data rates [72]. This section summarizes references that consider this problem. The approaches include channel assignment, transmit power control and

dynamic spectrum access. Specifically, a node can change its channel dynamically to maximize the opportunity of successful transmissions, where a node can either be an AP or a station.

The following works solve the interference problem through channel assignment and transmit power control. Reference [66] designs a Q-learning-based resource allocation algorithm. Each AP serves one single user. An AP learns to assign itself a channel and tune its transmit power independently. The objective is to maximize the SINR of users to improve their throughput. The state observed by an agent contains the SINR value of users. The action taken by an agent is to assign a channel and transmit power level. The reward received by an agent corresponds to the experienced throughput. Reference [67] proposes an actor-critic-based power control algorithm for packetized data links. There are several transmitters with finite queues. The algorithm is decentralized and each transmitter runs an agent that helps configure its transmit power based on its state. Specifically, its state is defined as (i) the queue length, and (ii) the corresponding receiver's SINR. The reward received by an agent corresponds to whether there are packet overflows or transmission failures; both of which result in a negative reward.

Reference [68] proposes another power control algorithm for APs. Each AP serves a single user and runs an RL agent. A central controller takes charge of computation and helps coordinate APs. The authors consider fading and shadowing effects. The aim is to (i) guarantee data rate fairness between APs, and (ii) maximize the throughput of APs. To guarantee fairness, APs with low throughput have a priority to use high transmit power. The state observed by an agent corresponds to the current transmit power level of an AP, the interference from its neighbors and the interference caused to its neighbors. The action taken by an agent is to allocate a transmit power level. The reward received by an agent relates to the throughput of the AP and the amount of capacity dropped due to its interference caused to its neighbors. Compared to [67] and [66], the agents in [68] works cooperatively. Specifically, they are aware of the interference caused to their neighbors. Also, an

agent receives a low reward if causes interference to its neighbors.

Reference [69] aims to optimize channel bandwidth, guard interval size, Modulation and Coding Scheme (MCS) as well as the frame aggregation level of an AP. The algorithm is decentralized and each AP runs an agent. The state observed by an agent corresponds to the status of these four parameters and the SINR value of users. The reward received by an agent refers to the bit error rate and frame error rate. The agent adjusts the said parameter based on the current state to minimize the bit error rate and frame error rate.

References [70] and [71] employ dynamic spectrum access approach to eliminate collision. The aim is to find a strategy for each node to access/sense the channel in order to maximize the number of successful transmissions. Each node runs an agent and decisions are made in a decentralized manner. Both works model channel access as a Partially Observable Markov Decision Process (POMDP). In [70], an agent observes the status of past transmissions, channels used and channel capacity. The said information is call memories. The authors employ a Long Short-Term Memory (LSTM) layer to store memories. An agent takes an action to pick a channel to transmit or remain silent. The reward received by an agent is a function of the total number of successful transmissions. In [71], the authors adopt a DQN module at both the sender and receiver. An agent observes the channel quality over a prolonged time period. The channel quality can only be good or bad, which is random. A sender and receiver can only sense/transmit on one channel in each slot. The reward received by an agent is related to whether there is a transmission success. A transmission failure is caused by poor channel quality, which leads to a negative reward. If a transmitter and a receiver are on a different channel, there will also be a failure. The DQN will be re-trained to find another policy if the agent receives a negative reward that exceeds a pre-designed threshold.

2.4 Discussion

In summary, this chapter has summarized prior works that consider:

- *Channel bonding*, which aims to reduce interference among nodes when they are using bonded channels, or to determine the bandwidth to nodes in order to meet their traffic demands.

- *Transmit power control*, where past works have considered employing transmit power control as well as channel assignment to increase the energy efficiency and capacity of a wireless system. Alternatively, they also consider reducing energy consumption and the interference among neighboring nodes.

- *Wireless charging*, where in future Wi-Fi systems, an AP/base station may be tasked with charging RF-energy harvesting nodes/users. The key problem is how to guarantee RF-energy harvesting devices have sufficient energy to achieve their sensing tasks, while not affecting data communications of legacy users.

- *Reinforcement learning*, where an agent learns to operate in cognitive radio, cellular, satellite or Wi-Fi systems. Prior works aim to mitigate interference or increase the data rate and energy efficiency of base stations/users.

However, existing works have the following gaps:

- Past channel bonding works such as [27–29] aim to maximize throughput or capacity. This means they do not consider traffic or assume saturated traffic at nodes. By contrast, this thesis considers time-varying traffic at APs. In works that consider traffic load, the authors employ integer/non-linear/chance-constraint programming or convex optimization; e.g., [15, 19, 26, 33, 34, 73]. These approaches are centralized, involve coordination between APs, and require perfect or global information. Critically, they are optimized for a specific scenario or nominal traffic demand. In contrast, the solutions proposed

in Chapters 3 and 4 do not have the said limitations. In addition, current traffic-aware channel bonding algorithms such as [19, 20, 33, 34] only consider a given amount of traffic load or assume APs know their traffic demands beforehand. For example, the work in [20, 29, 30] does not consider traffic variation when bonding channels. Their objective is for the network to reach Nash Equilibrium for a specific traffic load condition/configuration. By contrast, the solutions proposed in Chapters 3 and 4 are capable of adapting to random traffic demands. This thesis also considers the impact of random channel conditions on capacity and interference experienced by an AP. Works such as [19, 35] do not consider random channel gains between APs and users. They assume the capacity of a single channel is constant. Moreover, the said works do not consider partially overlapping channels. As noted in [21], these channels have the potential to increase the throughput of a WLAN. Lastly, no past works have considered energy efficiency optimization when APs use a bonded channel. Although the closest work in [31] considers the EE of APs, it does not consider traffic variations. Also, the solution in [31] does not have any learning ability.

- The work in this thesis is fundamentally different to those that consider both data communication and RF-charging in Wi-Fi systems. In references [10, 12, 37–39], APs or base stations have no EH capability. This thesis considers the challenge that an AP has no knowledge of solar energy arrivals. This is critical because an AP's energy level is bounded by its battery level and energy arrivals. This means an AP may experience energy outages in the future. Also, this thesis aims to control the transmit power at an AP with stochastic energy arrivals in order to satisfy *both* RF-energy devices and legacy Wi-Fi users. Another distinction to prior works is that this thesis considers imperfect CSI to IoT devices and causal knowledge of energy arrivals. Moreover, they use mathematical optimization techniques that require non-causal information.

On the other hand, this thesis employs machine learning approaches that allow an AP to adapt to its historical CSI and energy arrivals.

- Prior works on transmit power control, such as [43, 46–48], do not consider channel bonding. Specifically, their solution assigns a single channel to an AP in order to minimize interference or increase energy efficiency. Their solution, such as MILP and convex optimization, does not have any learning ability. Last, they do not consider traffic variations at APs. Another fundamental difference is that they only consider either data users or RF-energy harvesting devices.

- Numerous works have considered applying RL to improve the throughput or energy efficiency of Wi-Fi networks; e.g., [66–71]. However, they do not consider energy efficiency maximization. Specifically, no one has applied RL to assign partially overlapping channels to APs in order to meet their time-varying traffic demands while maximizing their energy efficiency. In addition, their system does include RF-energy harvesting devices. Lastly, they do not consider APs with energy harvesting capability.

Chapter 3

Channel Bonding with Random Traffic Demands

As shown in Chapter 2, a number of solutions aim to assign a bonded channel to APs. These solutions, however, are centralized, involve coordination between APs, and require perfect or global information. Critically, they are optimized for a given scenario or nominal traffic demand; e.g., [20, 33, 34] and [19]. In contrast, the proposed DRL-based algorithm in this chapter does not have the said limitations. Lastly, prior works on adaptive channel bonding do not consider partially overlapping channels. As noted in [21], assigning partially overlapping channels to APs has the potential to increase their throughput. This is because the use of partially overlapping channels helps increase spectrum efficiency in some cases. For example, two APs are sufficiently far apart.

Motivated by their limitations, this chapter proposes a novel DRL-based solution that assigns a bonded channel to APs given the varying number of users and interference that stems from using partially overlapping channels. In particular, it learns the optimal channel bonding policy that meets the time-varying traffic demands of APs while minimizing the interference among APs. This chapter formulates the

channel bonding problem as a Markov Decision Process (MDP). Next, it applies a DRL or Deep Q-network (DQN) [17] to determine the optimal policy used by APs to maximize their reward.

The remainder of this chapter has the following structure. Section 3.1 models a Wi-Fi network and its user distribution. Section 3.2 describes the DRL-based channel bonding algorithm. Section 3.3 presents numerical results, and Section 3.4 concludes this chapter.

3.1 System Architecture

There is an IEEE 802.11 WLAN with N APs. Each AP and its associated users form a BSS. Each BSS is defined by its AP; this means there are N BSSs. APs are assumed to be aware of each other's traffic demands and the number of associated users. This information can be obtained via the IEEE 802.11k standard, which provides an AP with the ability to measure the traffic load of an AP. APs can exchange this information directly via a controller [74]. There are M channels that are either marked as primary or secondary. Each channel has a bandwidth of 20 MHz and offers C Mbps. All APs have been assigned a *primary* channel by an existing channel assignment algorithm; see [75] for a survey on channel assignment algorithms. Critically, APs are able to aggregate their primary and secondary channels to form a bonded channel to potentially offer higher data rates to devices. Time is divided into slots with duration $\tau = 1$ hour. The set of time slots is $\mathcal{T} = \{1, 2, \ldots, T\}$, where there are in total T slots.

In each time slot, an AP's traffic demand is proportional to the number of associated users, where each user has a fixed traffic demand. Henceforth, the following sections will present the model that governs the random user arrivals at each AP, interference between channels, and also how to calculate the reward when an AP meets the traffic demand of all users in each time slot. Table 3.1 summarizes the nomenclature.

Table 3.1: Common Notations

Notation	***Description***
N	Number of APs
M	Number of channels
C	Capacity of a single channel
τ	Duration of one slot
γ_i	Exogeneous arrival rate of open class users
p_{ji}	Switching probability of open class users
λ_i	Aggregated arrival rate of open class users
N_c	Total number of closed class users in the system
v_i	Switching probability of closed class users
ρ_{i_o}	Aggregated traffic load due to open class users
ρ_{i_c}	Aggregated traffic load of closed class users
U_{i_o}	Number of open class users in AP i
U_{i_o}	Number of open class users in AP i
U_i^t	Total number of users in AP i
Γ_i	Set of APs that interfere with AP i
$o_{a,b}$	Channel overlap factor
c_i^t	Total capacity of a BSS
$O^t(i,z)$	Amount of channel overlap between BSS i and z
ζ_i^t	Channels assigned to AP i
c_i^t	Total capacity assigned to AP i
D	Traffic demand of a user
S_t^i	It indicates whether AP i satisfies its traffic demands at time slot t
I_t^i	It indicates whether AP i experiences the maximum interference at time slot t

3.1.1 User Arrival Model

The number of users associated to each AP in each time slot is given by the mixed queueing model by Chen et al. [2]. The said model is depicted in Fig. 3.1. There are two classes of users depending on their mobility properties; namely, *open* and *closed*. The open class has an uncertain number of users who arrive exogenously. The behavior of open class users in BSS i can be modeled as an M/G/∞ queue, where their arrival and departure is modeled as a Poisson process. An open class user enters BSS i via two paths: i) exogeneous arrivals with rate γ_i, and ii) from another BSS j to BSS i with probability p_{ji}. The aggregated arrival rate λ_i of BSS i is defined as $\lambda_i = \gamma_i + \sum_{j,i\neq j}^{N} \lambda_j p_{ji}$. The aggregated traffic load due to open class

users of BSS i is calculated as $\rho_{i_o} = \frac{\lambda_i}{\mu_i}$, where μ_i is the service rate of AP i. After receiving service, they leave the system. The expected residence time/service time of users in BSS i is denoted as $\frac{1}{\mu_i}$. Let U_{i_o} be the number of open class clients in BSS i. The probability that there are u_i open class users in BSS i is calculated as per,

$$P(U_{i_o} = u_i) = e^{-\rho_{i_o}} \frac{(\rho_{i_o})^{u_i}}{u_i!}. \tag{3.1}$$

Different from open class users, closed class users do not exit the system and only switch between different BSSs. The total number of closed class users is fixed, which is denoted as N_c. Each closed class user stays in BSS i with probability v_i. Let ρ_{i_c} be the total traffic load of closed class users in BSS i, where $\rho_{i_c} = N_c \times v_i$. Let U_{i_c} be the total number of closed class users associated to AP i. As the behavior of users is independent, the binomial distribution with parameter v_i is used to determine the probability that AP i has u_i closed class users. Formally,

$$P(U_{i_c} = u_i) = \binom{N_c}{u_i} (v_i)^{u_i} (1 - v_i)^{M - u_i}. \tag{3.2}$$

It is well-known that the binomial distribution can be approximated by a Poisson distribution with parameter ρ_{i_c}. Let U_i be the total number of users in BSS i, where this quantity is calculated as the convolution of the Poisson distribution that represents the number of open and closed class users: $U_i = U_{o_i} + U_{c_i}$. As the sum of independent Poisson distributions yields another Poisson distribution, and thus,

$$P(U_i = u_i) = e^{-\rho_i} \frac{(\rho_i)^{u_i}}{u_i!}. \tag{3.3}$$

Here, ρ_i includes the traffic load of both open and closed users in BSS i, and it is defined as $\rho_i = \rho_{i_o} + \rho_{i_c}$. Equation (3.3) thus can be used to determine the probability that the total number of users in BSS i is equal to u_i. In the sections to follow, symbol U_i^t denotes the number of users in BSS i at time slot t; the random value U_i^t is governed by (3.3).

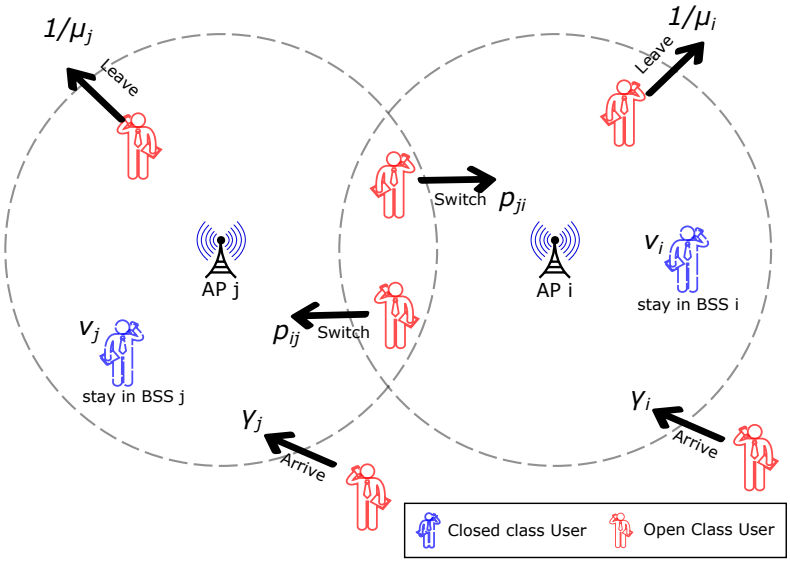

Figure 3.1: An illustration of user arrivals and departures in a WLAN.

3.1.2 Interference Model

A channel may experience interference from overlapping channels. The degree of interference between channel a and b is proportional to their so called overlap factor $o_{a,b}$; see also [76] for the definition of $o_{a,b}$. It represents the proportion of overlap in frequency between channel a and b, which increases as the distance between channel a and b decreases. For example, in [76], if $a = b$, then $o_{a,b} = 1$. If the channel is one apart, say $a = 1$ and $b = 2$, then $o_{a,b} = 0.77$ or $o_{a,b} = 0.96$.

Two BSSs that are within the carrier sense range of each other, i.e., they interfere with one another if they operate on the same channel, are called overlapping BSSs (OBSSs); similarly, two APs that interfere are called neighboring APs. Let Γ_i be the set of APs that interfere with AP i, meaning if AP i overlaps with AP j, k and l, then $\Gamma_i = \{j, k, l\}$. Note, the relationship between any two BSSs is decided when a WiFi network is built, meaning Γ_i is fixed

Define ζ_i^t to be a set containing channels assigned to AP i at time slot t. The total capacity of a BSS is determined by how many channels its AP uses, and it is defined as $c_i^t = |\zeta_i^t| \times C$, where C is the capacity of a single channel.

Recall that users from two OBSSs may interfere if their AP is assigned overlapping channel(s). Hence, the amount of bandwidth received by each user is propor-

tional to the number of users on the same channel and overlapping channels. For example, consider two neighboring APs: AP-a and AP-b. Each with five and ten users, respectively. If both APs are assigned the same channel C_1, then this channel has in total 15 users. However, if both APs have a non-overlapping channel, say C_2 and C_3, then AP-a has five users contending on channel C_2, and AP-b with channel C_3 has ten users. If the overlap between channel C_2 and C_3 is at 50%, then AP-a has $5 + 0.5(10) = 10$ users, and AP-b has $10 + 0.5(5) = 12.5$ users.

Let the function $O^t(i, z)$ return a weight in the range $[0, 1]$ that represents the amount of channel overlap between BSS i and z at time t; e.g., if AP i and z use the same channel in time slot t, then $O^t(i, z) = 1$, meaning all the users associated to AP i and z are treated as being on the same channel. Define $\hat{U}_t^i(.)$ to be a function that returns the total number of users operating on the channel(s), i.e., ζ_i^t, assigned to AP i in time slot t. Specifically, it is calculates as,

$$\hat{U}_t^i(\zeta_i^t, \Gamma_i) = U_i^t + \sum_{z \in \Gamma_i} O^t(i, z) U_z^t, \tag{3.4}$$

where

$$O^t(i, z) = \min\left(1, \sum_{a \in \zeta_i^t, b \in \zeta_z^t} o_{a,b}\right). \tag{3.5}$$

Here, the number of users associated to an AP U_i^t is a random number drawn from the user model in Section 3.1.1.

3.1.3 User Traffic Demand

This section models the aggregated user traffic demand of APs, which depends on both the random number of users associated to an APs and the channels used by APs. Without loss of generality, all users are assumed to have a fixed demand of D Mbps. Let $\hat{d}_t^i(\zeta_i^t, \Gamma_i)$ be the aggregated traffic demand of $\hat{U}_t^i(\zeta_i^t, \Gamma_i)$ users, which is expressed as $\hat{d}_t^i(\zeta_i^t, \Gamma_i) = D \times \hat{U}_t^i(\zeta_i^t, \Gamma_i)$. For AP i, if its total capacity c_i^t is greater than $\hat{d}_t^i(.)$ in slot t, then the traffic demand of AP i is considered to be satisfied in

slot t. If AP i is able to satisfy its user demands at time t, then $S_i^t = 1$; otherwise, $S_i^t = 0$. In addition, define I_i^t to indicate whether AP i experiences the maximum interference. That is, its value is $I_i^t = 0$ when for any given AP z in Γ_i, then $O^t(i, z) = 1$; otherwise, $I_i^t = 1$.

3.1.4 Problem

Now, the problem at hand is to determine a channel bonding policy that maximizes the average time slots in which APs satisfy their user demands while minimizing their chance to experience the maximum possible interference. In other words, the aim to allocate only sufficient capacity to APs corresponding to their traffic load, rather than maximize their total capacity. Note, this means that the throughput of APs will depend on their user traffic demands. Specifically, a policy that maximizes the following quantity:

$$\lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{T-1}\sum_{i=0}^{N} S_t^i I_i^t\right]. \tag{3.6}$$

3.2 Solution

The problem at hand is re-formulated as an MDP. Then, this section introduces a DRL solution, its architecture, update policy and channel bonding strategy. Lastly, the MDP is solved by the DRL framework.

It is worth noting that the underlying user mobility model, see (3.3), follows a Poisson process. Hence, the use arrival to an AP varies slot by slot independently, and thus satisfies the Markov property. The notations that appear in this section are listed in Table 3.2.

3.2.1 Markov Decision Process

An MDP is defined by the tuple $(S, A, P(s_{t+1}|s_t, a_t), R(s_{t+1}|s_t, a_t))$. Here, S is a set of states. The state at time slot t is denoted as s_t, where $s_t \in S$. The set A

Table 3.2: Common Notations

Notation	***Description***
s_t	The state observed by agent i at time t
a_t	The action taken by agent i at time t
$P(s_{t+1}\|s_t, a_t)$	State transition function
$r(s_{t+1}\|s_t, a_t)$	The reward after taking action a_t at state s_t
$Q(s_t, a_t)$	Q-value for $s = s_t$ and $a = a_t$
α	Learning rate
γ	Reward discount factor
$\hat{Q}(r_t, s_{t+1}, a_{t+1})$	Target Q-value
θ	Prediction network
θ'	Target network
$L(\theta)$	Loss function
T_l	Interval of updating the prediction network θ
T_r	Interval of replacing the parameters the target network θ
$\mathcal{D}_i$	Memory buffer of agent i
ϵ	Exploration rate
d_t^i	Aggregated traffic demand of the users associated to AP i at time slot t
$P_{d_t^i}$	Prediction for the traffic demand d_t^i associated to AP i
$\overline{d_t^i}$	Average traffic demand associated to AP i over the past ten slots at time slot t
Γ_i	Set of APs that potentially interfere with AP i

contains a group of feasible actions. The action taken at time t is denoted as a_t, where $a_t \in A$. The state transition function $P(s_{t+1}|s_t, a_t)$ indicates the probability that the current state s_t moves to the next state s_{t+1} after taking action a_t. The reward function $R(s_{t+1}|s_t, a_t)$ returns the attained reward r_{t+1} after taking action a_t at state s_t. Let π define the policy taken by an agent. Specifically, the function $\pi(s_t)$ returns the action taken by the agent in state s_t, and π^* denotes the optimal policy. Given an MDP, the problem at hand is to seek the optimal policy π^* that maximizes the following long-term cumulative reward:

$$\mathbb{E}\left[\sum_{t=0}^{+\infty} R(s_{t+1}|s_t, \pi(s_t))\right]. \tag{3.7}$$

In the MDP model in question, agents will observe their traffic demand, and take an action; i.e., whether they should use bonded channels. They then obtain a reward depending on whether the taken action satisfies their traffic demand. As traffic demand is *random*, agents do not know how traffic demands change and what reward will be given by the environment. This means the abovementioned MDP is model-free as its transition probability $P(s_{t+1}|s_t, a_t)$ is unknown. Therefore, standard approaches such as value or policy iteration cannot be used to compute policy π^*. To this end, the next section will introduce a DRL approach to find the optimal policy for a model-free MDP [77].

3.2.2 Reinforcement Learning

The Q-leaning [78] method can be used by an agent to learn the state-action-value or Q-value over time. In other words, it enables an agent to learn the expected discounted cumulative reward for an action taken at a given state, which is then used to update the corresponding Q-value of state s_t and action a_t as per Bellman's equation,

$$Q_{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\hat{Q}(r_t, s_{t+1}, a_{t+1}) - Q(s_t, a_t)), \tag{3.8}$$

where

$$\hat{Q}(r_t, s_{t+1}, a_{t+1}) = r(s_t, a_t) + \gamma \max Q(s_{t+1}, a_{t+1}). \tag{3.9}$$

In the foregone expression, the variable $\alpha \in (0, 1]$ is the learning rate and $\gamma \in [0, 1]$ is the discount factor. The target Q-value and predicted Q-value is defined as $\hat{Q}(r_t, s_{t+1}, a_{t+1})$ and $Q(s_t, a_t)$, respectively. It is widely known that Q-Learning converges if an agent visits all states sufficiently often [78].

Conventional Q-learning has two issues when the problem at hand has a large state space. First, an agent may fail to update most states. Another issue is that the Q-table will become too large, making searching and storage impractical [70].

To overcome these issues, this chapter applies a DQN framework [17]. The DQN framework is an extension of tabular Q-learning. It uses two neural networks to approximate Q-tables, which allows them to handle large and even continuous state space. It has three key features: update strategy, memory replay strategy and action selection strategy. These features will be explained in the next sections.

3.2.2.1 Update strategy

The aim is to minimize the average difference between the target Q-value $\hat{Q}(r_t, s_{t+1}, a_{t+1})$ and predicted Q-value $Q(s_t, a_t)$. To achieve this, the DQN framework contains two neural networks: a prediction network θ and a target network θ', which are used to compute predicted and target Q-values. The mean square difference between these two values is defined as the loss and is denoted as $L(\theta)$. When training a DQN, the goal is to minimize the following quantity:

$$L(\theta) = \min \mathbb{E}[(\hat{Q}(r_t, s_{t+1}, a_{t+1}, \theta') - Q(s_t, a_t, \theta))^2]. \tag{3.10}$$

In every T_l slots, the prediction network θ updates its parameters according to Equ. 3.10, where the parameters are the weight and bias of neurons. The target network θ' has the same topology as the prediction network θ but has a different update policy. Specifically, the target network θ' is updated less frequently than the prediction network θ. In every T_r slots, where $T_r \gg T_l$, the parameters of the prediction network θ become those of the target network θ'.

3.2.2.2 Memory replay strategy

To increase sample utilization and training convergence, DQN applies a memory replay strategy. Each agent maintains a memory buffer $\mathcal{D}_i$ to store the visited transition pair (s_t, a_t, s_{t+1}, r_t). The size of this memory buffer is denoted as $N_{\mathcal{D}}$. At each slot, a new transition pair will be stacked into memory. The oldest transition pair will be deleted from $\mathcal{D}_i$ once the memory buffer $\mathcal{D}_i$ is full. The agent periodically

samples a mini-batch of memories to minimize the loss $L(\theta)$. This is an application of Stochastic Gradient Descent (SGD) with learning rate α [22].

To speed up convergence, the data in the memory buffer $\mathcal{D}_i$ needs to be pre-processed. Specifically, if $\mathcal{D}_i$ contains the historical traffic demands of AP i, then to normalize the traffic demands, the following Z-score method is applied:

$$x' = \frac{x - \sigma}{\mu}, \tag{3.11}$$

where x and x' are the value of the original and normalized data, respectively. The term μ and σ correspond to the average and standard deviation of the original data in $\mathcal{D}_i$.

3.2.2.3 Action selection strategy

Fig. 3.2 illustrates how an agent chooses an action for a given state. At time slot t, the prediction network θ is fed the state s_t and it will output the predicted Q-value $Q(s_t, a_t, \theta)$ for each action. An agent then selects an action according to the maximum predicted Q-value $Q(s_t, a_t, \theta)$. To ensure sufficient exploration of the action space, agents adopt the ϵ-greedy policy to choose an action. Specifically, an agent chooses the maximum Q-value action with probability $(1 - \epsilon)$ or it randomly selects an action with probability ϵ, where ϵ is the exploration rate. Let ϵ_0=1 and ϵ_T be respectively the initial and final exploration rate. At time t, the exploration rate ϵ is calculated as per,

$$\epsilon_t = \max\left[1, \epsilon_T + \frac{\epsilon_{inc}}{(t+1)^2}\right], \tag{3.12}$$

where ϵ_{inc} is the increment rate of ϵ and has range $[10^3, 10^6]$.

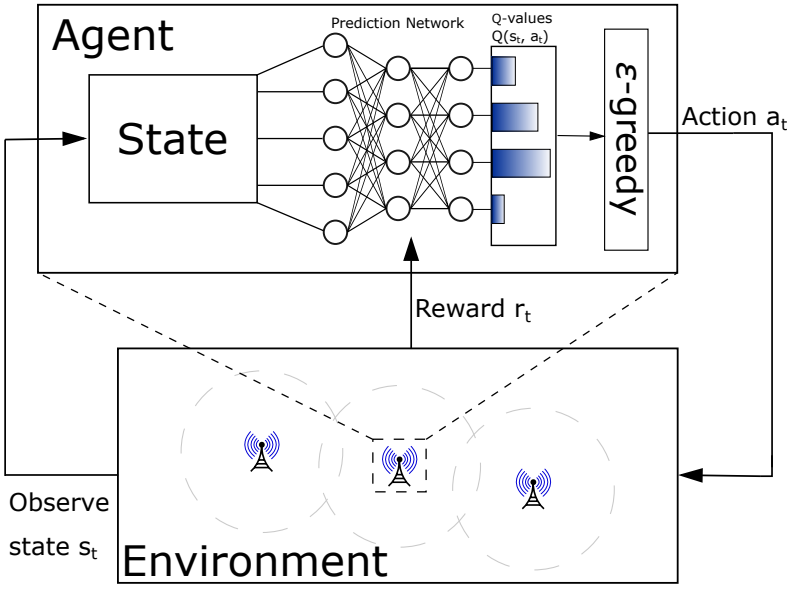

Figure 3.2: The action selection strategy of an agent.

3.2.3 Instantiation

This section shows how APs use DQN to bond channels. Each AP runs a DQN agent, and observes its own traffic demand and that of its neighbors. The traffic traffic demand that belongs to AP i at time slot t is denoted as d_t^i, which is calculated as $d_t^i = U_i^t \times D$. There are two state definitions, denoted as **State-1** and **State-2**. For **State-1**, the state observed by AP i corresponds to the traffic demand in the *current* slot and the predicted traffic $P_{d_t^i}$ for the next slot and that of its neighbors. **State-1** is defined as,

$$s_t^i = \{(d_t^a, P_{d_t^a}), \ldots, (d_t^i, P_{d_t^i}), \ldots, (d_t^z, P_{d_t^z})\}, \tag{3.13}$$

where d_t^i is the total user traffic demand of AP i at time t, $P_{d_t^i}$ is the probability that the traffic demand of the next slot d_{t+1}^i is greater than d_t^i. Also, a to z, except i, denote APs in Γ_i. The predicted traffic $P_{d_t^i}$ of AP i can be calculated using historical traffic demands stored in the memory buffer $\mathcal{D}_i$. For example, if $d_t^i = 200$ Mbps, then $P_{d_t^i}$ is calculated as the number of times that the traffic demand is greater than 200 Mbps in the memory buffer divided by the buffer size $N_{\mathcal{D}}$. This is known as the cumulative distribution probability of traffic demands.

The second state definition, namely **State-2**, employed by AP i corresponds to its average traffic, and that of its neighbors over the *past ten* slots and at time slot t. This is defined as

$$s_t^i = \{\overline{d_t^a}, \ldots, \overline{d_t^i}, \ldots, \overline{d_t^z}\}, \tag{3.14}$$

where $\overline{d_t^i} = \frac{d_{t-9}^i + \ldots + d_t^i}{10}$.

We can see that size of state space is proportional to the number of neighbors of AP i. For **State-1**, its dimension is $2 \times |\Gamma_i|$. For **State-2**, its dimension is $|\Gamma_i|$.

Now define DQN-1 and DQN-2; they are fed with **State-1** and **State-2**, respectively. For a given AP i, its action space A_i contains all possible channel assignments. As an example, consider Fig. 3.3. As we can see, AP i is able to bond several secondary channels that are adjacent to its primary channel-2. The channel configuration set, also known as the action space A_i, contains all possible combinations of the primary channel and secondary channels of AP i. Note, a bonded channel must be consecutive or adjacent. In Fig. 3.3, there are six bonded channels for AP i, meaning the size of action space is also six. At time slot t, this AP selects to use one of channel configurations from A_i for time slot $t+1$ according to the action selection strategy. Then, its agent will gain a reward.

The reward r_t^i gained by AP i at time slot t corresponds to whether it is able to satisfy traffic demand and experiences the maximum interference. As defined in Section 3.1.4, $I_i^t = 1$ indicates AP i is *not* suffering the maximum interference. Also, indicator $S_t^i = 1$ represents that the traffic demand of AP i at slot t is satisfied. The reward $R(t, i)$ for AP i at time slot t is defined as,

$$R(t,i) = \begin{cases} 1, & S_t^i = 1, I_t^i = 1 \\ -0.0001, & Otherwise \end{cases} \tag{3.15}$$

Lastly, it is worth noting that DQN is trained in an offline manner, where choosing actions and updating the policy of a DQN are carried out separately. To be specific, an agent/AP only needs to observe the state s_t and choose an action a_t as

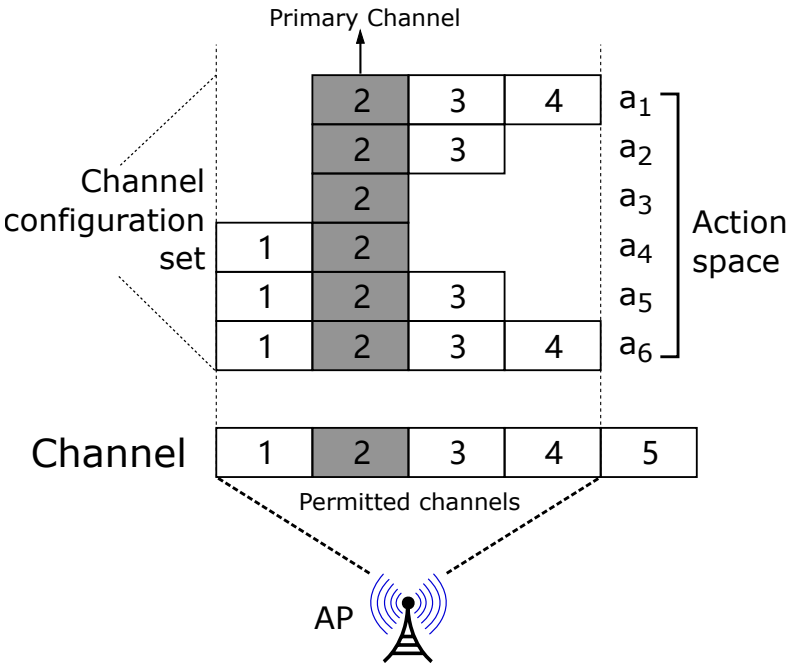

Figure 3.3: An illustration of the action space of an AP.

per the policy provided by a trained DQN. An AP can upload its historical state, action and reward to a controller[1] periodically. This information is then used to train the DQN, and once a new policy is ready, APs are installed with a new policy. It is worth noting that that information such as the number of associated users is available readily at each AP.

3.3 Evaluation

The proposed DRL algorithms are evaluated by Python 3.5 with TensorFlow 1.0 and MATLAB 2017. They are compared against the following algorithms/rules:

- **Tabular RL (TRL)**: This algorithm uses a table to store Q-values. The state is the same as that of DQN-2.
- **Widest Channel Only (WCO)**: Every AP bonds the maximum number of secondary channels. If an AP is able to bond a maximum of four channels, then it will always use four channels.

[1]Example WLANs controllers are those sold by vendors such as Cisco.

- **Primary Channel Only (PCO)**: APs do not use channel bonding and only use their primary channel.
- **Random**: Every AP uniformly chooses a channel configuration in each slot. For example, an AP either chooses to use its primary channel only, or both its primary and secondary channels.
- **Heuristic**: If an AP cannot satisfy its demand in the last time slot, it will increase its bandwidth by bonding an additional secondary channel. Otherwise, the AP will decrease its bandwidth by releasing one secondary channel. Note, APs always have their primary channel, and do not bond more than the available number of channels.

Note, these rules are as per the current WiFi networks rules. For example, PCO considers the case where APs do not bond channel. Random considers the case where channels are randomly occupied by APs. In each slot, APs take an action according to these rules, and then observe the resulting reward as per 3.15.

The following metrics are recorded for evaluations:

- **Fraction of satisfied demands**. The fraction of satisfied demands of a given AP i is recorded for every k time slots, where $k = 250$. This fraction corresponds to the number of satisfied time slots over k time slots.
- **Percentage of satisfied users**. For a given AP i, this is calculated as $\min(1, \frac{\hat{d}_t^i(.)}{c_i^t})$ for each time slot t.
- **Reward**. This is defined as the average reward gained by agent i; for every k time slots, it is calculated as $\frac{\sum_{t-k}^{t} r_t^i}{k}$.
- **Amount of maximum interference**. This is defined as the proportion of k time slots in which an AP experiences the maximum interference.
- **Channel usage**. This corresponds to the number of time slots in which an AP uses a given channel.

Table 3.3: Parameter values used in experiments. The bold parameters are similar to those reported in [2].

Model Parameters	***Values***
Number of APs N	10
Interference probability	60%
Number of channels M	10
Primary channels	1, 4, 7 and 10
Demand per user D	25 Mbps
Capacity of each channel C	200 Mbps
Maximum bonded channel	Up to four channels
Simulated time slot T	100,000 time slots
Overlap factor $o_{a,b}$ when $a = b$	1
Overlap factor $o_{a,b}$ when $\|a - b\| = 1$	0.5
Overlap factor $o_{a,b}$ when $\|a - b\| > 1$	0
Total number of closed class users N_c	441
Exogenous arrival rate γ_i	$\mathcal{U}[1, 5]$
User switching probability p_{ij}	$\frac{1}{N+1}$
Expected residence time $\frac{1}{\mu_i}$	$\mathcal{U}[1, 5]$
Fraction of time of stay ν_i	$\mathcal{U}[0, 017]$

Algorithm Parameters	***Values***
Learning rate α used by TRL	0.01
Learning rate α used by DQNs	0.007
Reward decay rate γ	0.2
Memory size $N_{\mathcal{D}}$	6,000 to 8,000
Mini-batch size N_{mb}	256
Updating interval for θ T_l	Every four slots
Replacing interval for θ' T_r	Every 400 slots
Replay start time slot	At slot 6,000
Activation function (Layer 1)	Sigmoid
Activation function (Layer 2)	ReLU
Initial exploration rate ϵ_0	1
Final exploration rate ϵ_T	0.05
Reward of for maximum interference	-0.0001
Reward of unsatisfactory demands	-0.0001
Reward of satisfactory demands	1
Results collection time	At slot 80,000
Observable history used by DQN-1	Last 10 slots

To train DQN agents, the system state is generated by the user arrival model in Section 3.2. The state, action and reward in each slot are then recorded in a memory buffer. After every four slots, the training data in the memory buffer is

then used to update the neural network that represents the value of each action; see Equ. 3.10. Agents are trained starting from the 6,000-th slot after they collect 6,000 memories. Also, after the 80,000-th slot, the well-trained DQN agents are tested. The average user satisfaction achieved by them is recorded and shown in Fig. 3.9 to 3.11. In particular, this section studies three aspects. First, it studies the percentage of satisfied users and channel usage of a dense WLAN with ten APs over time. The probability that any two APs are neighbors is 60%, meaning that each AP overlaps with six APs on average. This probability characterizes the density of a WLAN and it is fixed for a given WLAN. Second, it investigates changing traffic demands. Third, it studies six WLANs with different setups such as different numbers of APs, channels, AP neighbors, and demand per user. To eliminate the variance across different runs due to random experimental environment, the same random seed is applied for every run. Therefore, the comparison between different solutions is fair. Also, within one experiment run, average metrics are evaluated, e.g., reward over a very large time horizon $T = 100,000$. This is to eliminate the deviation caused by randomness accoss different slots. Table 3.3 lists the parameter values used in simulations.

Fig. 3.4, 3.6, 3.5 and 3.7 show the performance of the tested algorithms over 100,000 time slots. From Fig. 3.4, 3.6 and 3.5, we see that the performance of all non-RL algorithms does not increase because they do not improve their channel bonding policy. Referring to Fig. 3.4, the average fraction of satisfied demands for Heuristic, Random, WCO and PCO is approximately 0.6, 0.56, 0.51 and 0.42, respectively. By contrast, the average fraction of satisfied demands attained by DQNs increases from 0.56 to 0.82. They are able to learn the best action after training. Before the 10,000-th slot, as marked by the ellipse in Fig. 3.4, 3.6, 3.5 and 3.7, the performance of DQNs is no different to the Random method, where the fraction of satisfied demands is 0.85. This is because TRL and DQNs have yet to converge, meaning that the actions taken by TRL and DQN agents are arbitrary. After the 10,000-th time slot, from Fig. 3.4, 3.6 and 3.5 we can clearly observe an

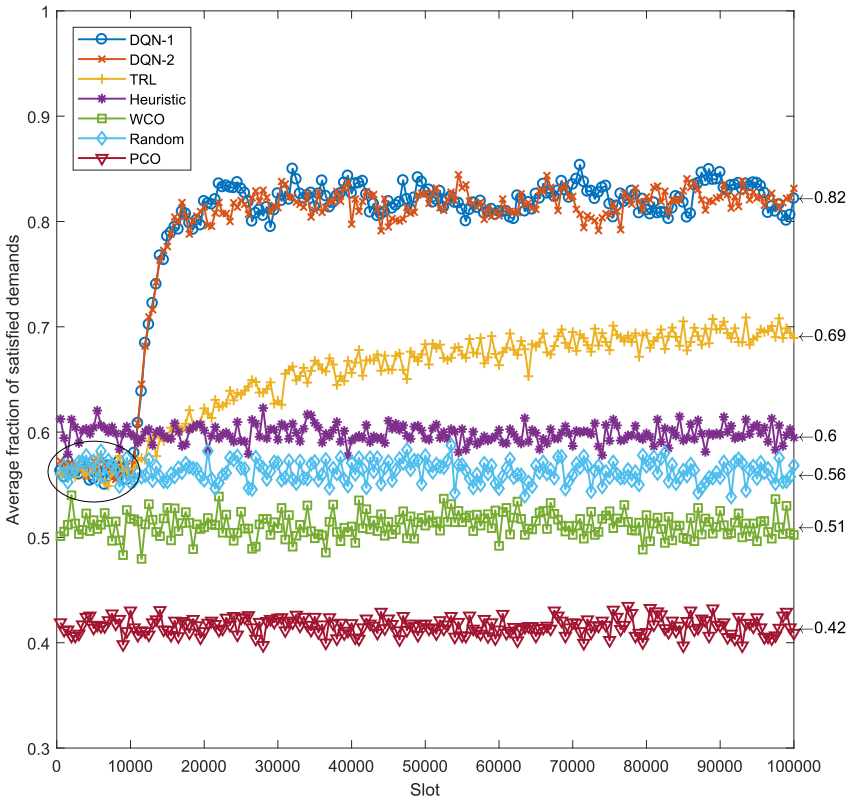

Figure 3.4: Elapsed time versus average fraction of satisfied demands.

upward trend in user satisfaction and reward for RL-based algorithms. One reason is that agents running DQNs begin to update their channel bonding policy according to their memory buffer $\mathcal{D}$. Also, as the exploration ϵ rate decreases, both TRL and DQN algorithms begin to exploit actions with the maximum Q-value. After time slot 30,000, the average fraction of satisfied demands, the percentage of satisfied users and the reward of DQN algorithms converge to around 0.82, 0.96 and 0.76, respectively. In other words, DQN algorithms improve user satisfaction by 30% to 70% as compared to non-RL algorithms.

We can see the difference in performance between DQN and TRL from Fig. 3.4, 3.6 and 3.5, where TRL's performance and convergence are inferior to DQN algorithms. TRL is the second-best performer among the tested algorithms. For instance, TRL achieves 0.1 to 0.3 more fraction of satisfied demands than non-RL algorithms, and its fraction of satisfied demands is less than that of DQNs by 18%. Referring to Fig. 3.5, we see that the average reward gained by agents plateaus at 0.51 after

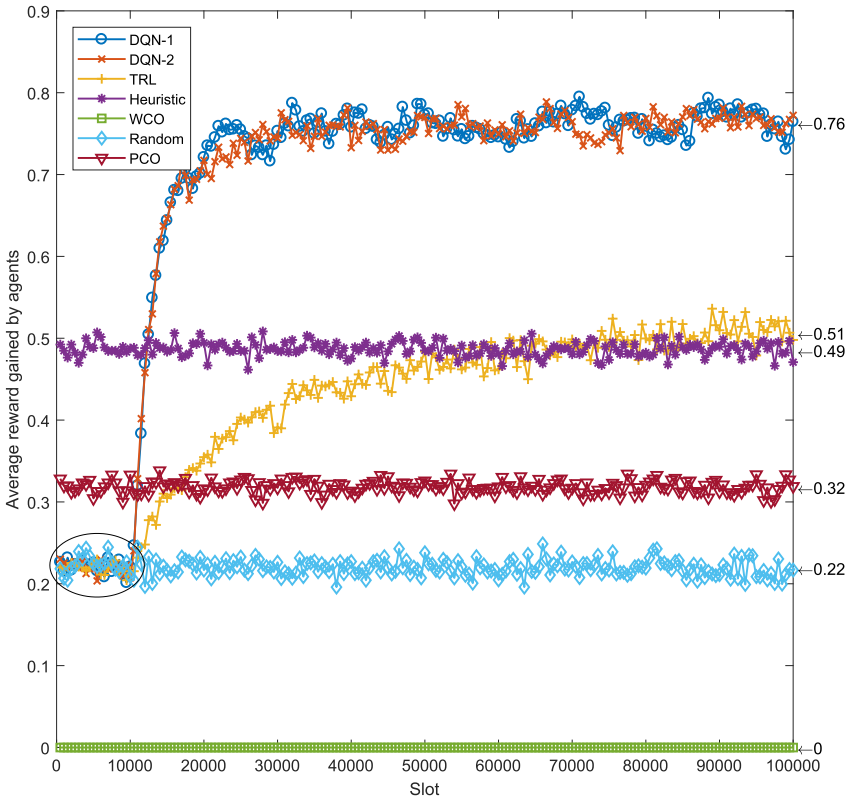

Figure 3.5: Elapsed time versus average reward gained by agents.

80,000 slots. As a comparison, DQN algorithms converge at slot 20,000. One reason is that DQNs are trained using a mini-batch of random samples, which minimizes the correlation between samples and thus accelerates convergence [17]. Another reason is that TRL is not suitable for handling a large state-action space. A limitation highlighted by [79] is that TRL only updates each state-action value separately, without any generalization. In particular, TRL repeatedly updates the Q-value of several frequently visited states, whereas, most state-action values are rarely updated [70]. Consequently, such an update policy has a slow convergence rate; indeed, it cannot converge to the optimal Q-values within the first 100,000 time slots. The state space is large; e.g., the observation of AP-2 has seven features and it contains more than 20^7 states. Here, one feature matches an input of a neural network. However, the TRL algorithm remains superior to non-RL algorithms by 20% to 50% in terms of user satisfaction and reward. From Fig. 3.5, DQN algorithms and TRL gain the highest and second-highest reward after training, achieving a user satisfaction of

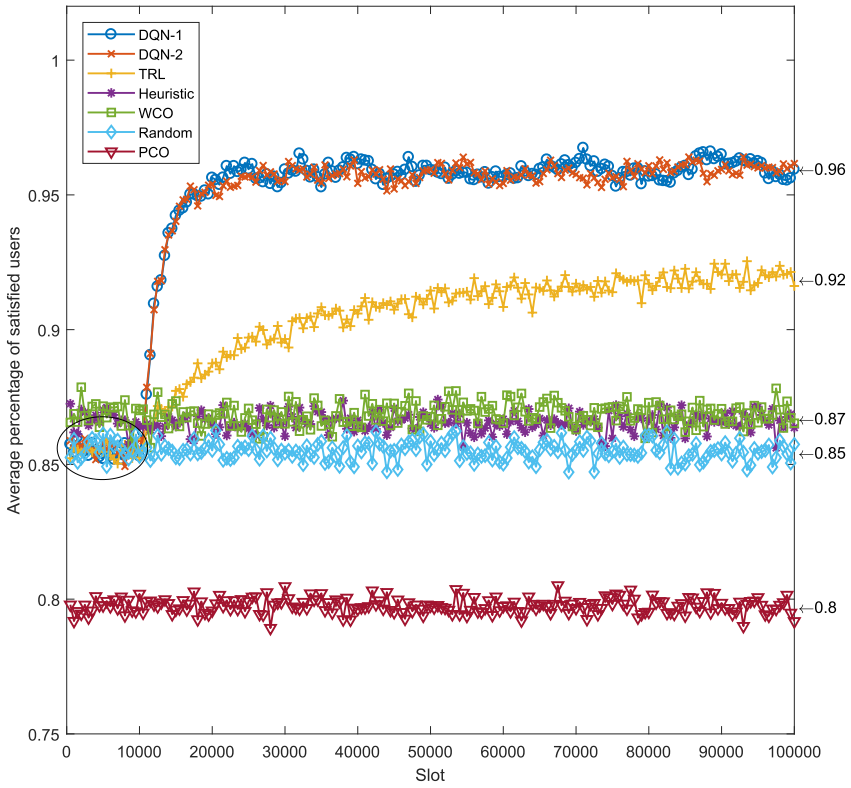

Figure 3.6: Elapsed time versus average percentage of satisfied users.

0.76 and 0.51 respectively. This is because these RL algorithms are able to learn the best action that maximizes the reward for each state. A noteworthy point with regards to Fig. 3.5 is that the reward of WCO is always -0.0001 because bonding four channels always leads to overlaps, which yields a negative reward.

Next, we study why TRL and DQN algorithms offer better performance by combining the results shown in Fig. 3.7 and 3.8. Fig. 3.8 illustrates the channel usage of DQNs, RTL, Heuristic and Random algorithms after 80,000 slots. The results for WCO and PCO are omitted because they do not change the channels assigned to APs. We can see that the Random method has the highest overall channel usage among the tested algorithms, meaning APs frequently use secondary channels. However, the average fraction of satisfied demands of Random is only 0.56; its percentage of satisfied users is 85%. Also, WCO always results in maximum interference among APs and thus its average fraction of satisfied demands is only 0.51; this is worse than the Random algorithm. These results show that even when APs use a bonded channel, they may experience poor performance because of excessive in-

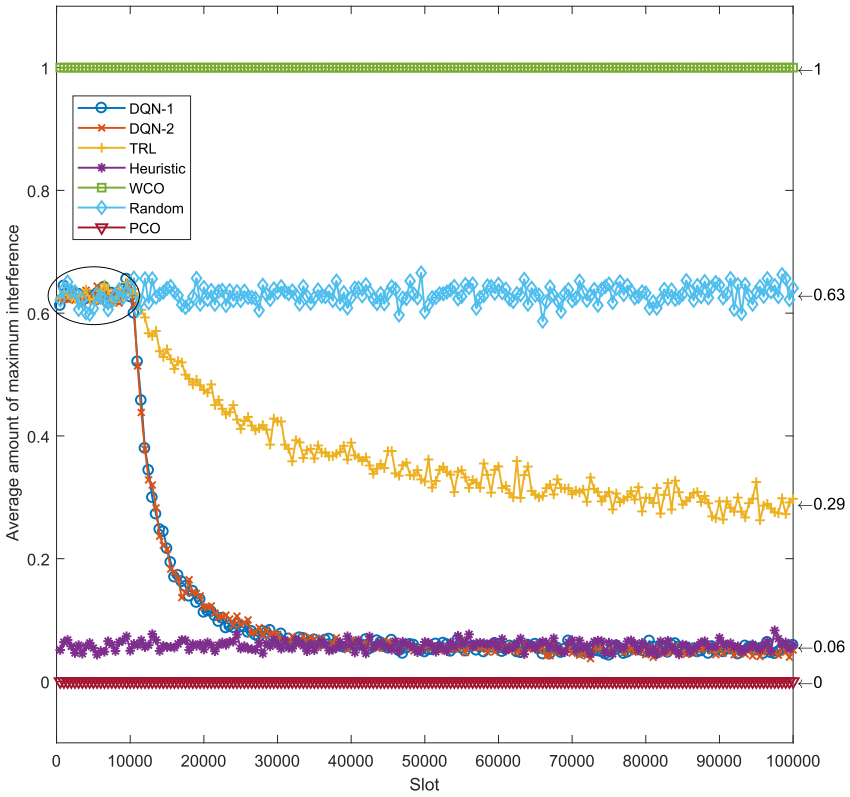

Figure 3.7: Elapsed time versus average amount of maximum interference.

terference. However, low interference does not mean high user satisfaction since there is a trade-off between bandwidth and interference. Heuristic does not have the same user satisfaction as DQN algorithms even though APs experience excessive interference. This is because Heuristic is too conservative when bonding channel, which is observable from Fig. 3.8 (d). For instance, channel-9, which is un-assigned, remains unused by AP-1 for around 10,000 time slots. We see that AP-1 uses only one channel during that time, which is inefficient.

Numerical results show that RL-based algorithms are able to balance between bandwidth and interference, especially when they receive a negative reward when they experience the maximum interference or an unsatisfied demand occurs. Fig. 3.7 shows that DQN algorithms learn to avoid interference after being trained. They decrease the average number of times in which APs experience the maximum interference from 0.63 to 0.06 upon convergence. Fig. 3.8 (a) and (b) further show that most neighboring APs do not interfere with each other; i.e., their channels do not overlap. For example, AP-1 and AP-2 are neighbors and their channels do not

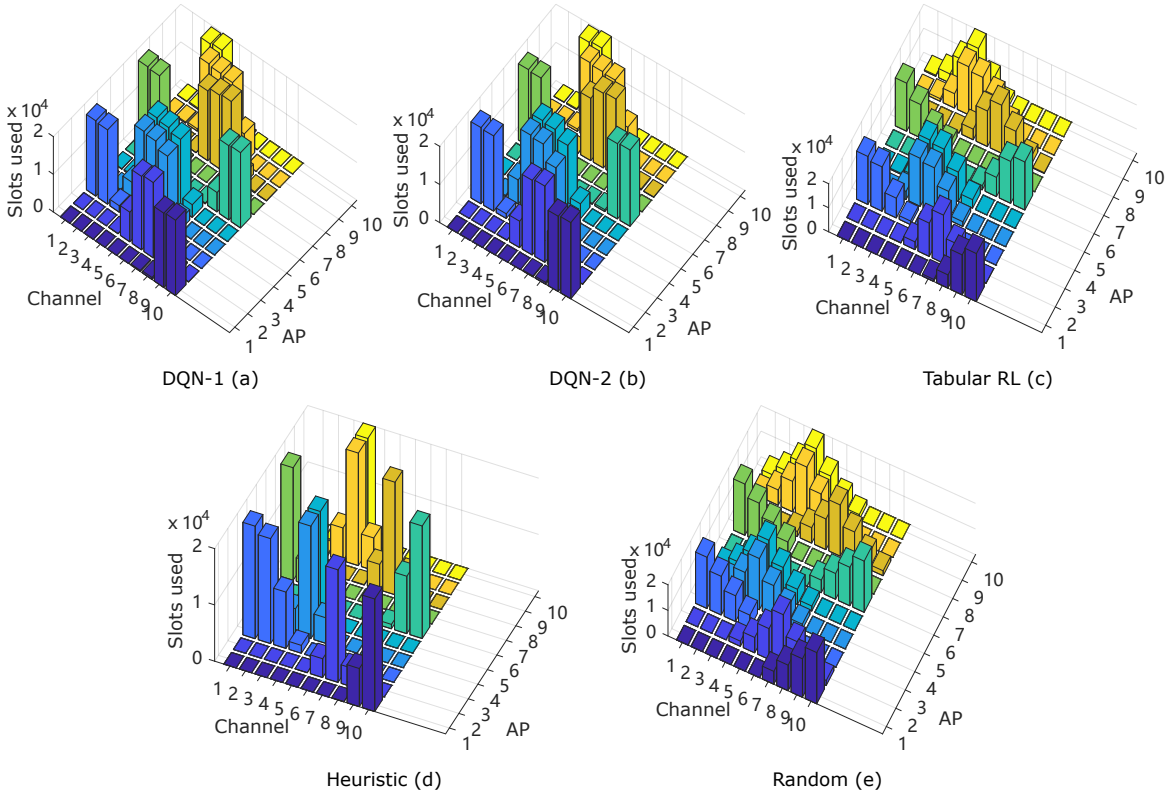

Figure 3.8: Channel usage of APs running DQN-1, DQN-2, TRL, Heuristic and Random.

overlap. Also, the results in Fig. 3.8 (a) and (b) show that DQNs agents are able to assign bonded channels to APs according to traffic demands. For example, as we can see from Fig. 3.8 (a), AP-9 bonded three channels for around 20,000 slots and bonded four channels for around 5,000 slots because it has more traffic demands than other APs. AP-1 and AP-7 only used two channels because they find that two channels are sufficient for their traffic demands, and only those two channels are interference-free. From Fig. 3.7, we see that when APs use TRL, the average amount of maximum interference experienced by them reduces from 0.63 to 0.29 upon convergence. Further, Fig. 3.8 (c) shows that APs using TRL may choose actions that lead to a high number of slots where APs experience the maximum interference. For example, we see that AP-6 and AP-8 experience the maximum interference on channel 8 for more than 8,000 slots. By contrast, for DQN-2, there is no interference between AP-6 and AP-8. Also, by comparing Fig. 3.8 (c) and (e), we find that the channel usage of AP-1, AP-6, AP-7 and AP-8 is no different to Ran-

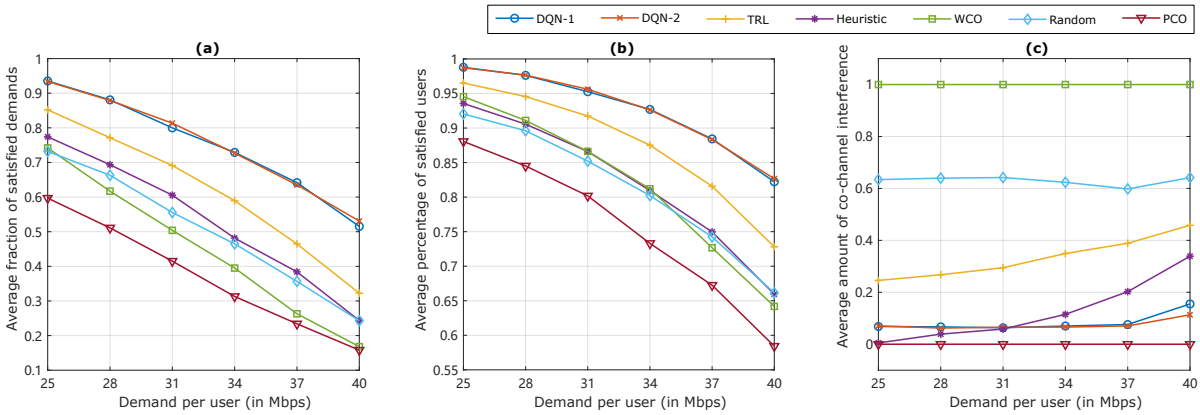

Figure 3.9: The impact of increasing traffic demand per user D.

dom, where these APs have one thing in common: they have many neighbors, and thus their state space is of high-dimension. In particular, after more than 100,000 slots, we find that the user satisfaction of these APs fluctuates sharply, meaning that the Q-table of these APs has yet to reach convergence. We see that for those APs with many neighbors, the TRL algorithm has difficulty converging to the optimal value.

From Fig. 3.4, 3.6, 3.5 and 3.7, we see that DQN-1 and DQN-2 have exactly the same performance. Also, Fig. 3.8 shows both DQN algorithms have executed the same action for 90% of the APs. This result is interesting because even though the state used by both DQN algorithms is different, they ended up with the same action.

Fig. 3.9 illustrates how traffic demand per user D impacts the average user satisfaction and the proportion of time APs experience the maximum interference. Fig. 3.9 (a) and (b) show that the user satisfaction decreases as the traffic demand per user D increases. To be specific, the user satisfaction of non-RL algorithms decreased by 60% to 80% on average. This is because APs require more secondary channels to satisfy traffic demands. However, excessively increasing the number of bonded channels reduces system performance, especially when traffic demands are high. This conclusion can be drawn from Fig. 3.9 (a), where the satisfaction probability of WCO exceeds PCO by 0.14 when $D = 25$ Mbps and the satisfaction

probability of WCO is the same with PCO when $D = 40$ Mbps. As a comparison, during high traffic demands, DQN algorithms continue to explore the widest bonded channel for APs while minimizing the number of slots where APs use the same channel(s). As shown in Fig. 3.9 (a), the average fraction of satisfied demands of DQN algorithms is 0.52 when $D = 40$ Mbps, which exceeds other algorithms by 66% to 188%. This is because APs do not experience high interference. As marked by the ellipse in Fig. 3.9 (c), the amount of maximum interference experienced by APs increases by only 0.1 when the traffic demand per user D is 40 Mbps. By contrast, the amount of maximum interference obtained by Heuristic, TRL and Random increased to 0.36, 0.44 and 0.62 respectively when $D = 40$ Mbps. The results show that DQN algorithms are able to improve system performance, especially in high traffic demands scenarios. We can see from Fig. 3.9 (a) that the difference in user satisfaction between DQN algorithms and other algorithms is around 0.1 to 0.3. However, when user demand is $D = 40$ Mbps, the difference in the fraction of satisfied demands ranges from 0.2 to 0.35, meaning that when $D = 40$ Mbps, the improvement in system performance provided by DQN algorithms is better than when $D = 25$. These results indicate that DQN algorithms are able to improve system performance in high traffic demands scenarios.

As shown in Fig. 3.9 (a) and (b), the average fraction of satisfied demands and percentage of satisfied users decline from 0.93 to 0.52 and from 0.99 to 0.82, respectively. The reason is that the total traffic demand will exceed the maximum channel capacity as traffic demand per user D increases. Consequently, some APs cannot find any channel assignment policy that will satisfy their heavy traffic demand. For example, assume every AP has a maximum capacity of $4 \times 200 = 800$ Mbps that is achieved by bonding four interference-free channels. However, the resulting capacity remains insufficient for some APs when $D = 40$ Mbps because these APs usually have more than 20 users. In addition, as APs use four bonded channels, there will be more interfering users, which reduces user satisfaction.

The next experiment investigates whether DQN is able to adapt to varying user

arrival rates. There are two sets of random user arrivals drawn from the traffic model using the parameters shown in Table 3.3. The first set of arrival rates is denoted as *Scenario-A*, and the second is denoted as *Scenario-B*. Then, there will be two simulations. In the *first* simulation, the arrival rate of each AP is as per Scenario-A before the 300,00-th slot. After the 300,00-th slot, the user arrival rate at APs will be as per Scenario-B. This means every AP will have a different user arrival rate after the 300,00-th slot. In the *second* simulation, for all time slots, the user arrival rate of each AP is drawn from Scenario-B only.

Fig. 3.10 shows the elapsed time slots versus the average fraction of satisfied demands. As marked by the dotted line in Fig. 3.10, DQN algorithms have an average user satisfaction of 0.61 in the second simulation. From Fig. 3.10, the first convergence is at around 25,000 slots, where the average fraction of satisfied demands is 0.81. At slot 30,000, the average fraction of satisfied demands drops dramatically to 0.5. One reason is that the channel bonding policy for scenario-A is not suitable for scenario-B, which leads to a mismatch between AP capacity and traffic demands. After switching to scenario-B, APs take a further 30,000 slots to adapt to new traffic demands, in which their average fraction of satisfied demands reaches 0.59. The difference to the first scenario is only 0.02, which has a user satisfaction of 0.61. This result indicates that APs running DQN algorithms are able to adapt to varying arrival rates even after convergence to scenario-A. This is because existing Q-values will be updated to reflect the traffic demands under scenario-A. Interestingly, we find that the second convergence took a longer time than the first convergence for scenario-A. The reason is that agents need a few slots to replace outdated experiences in the memory buffer $\mathcal{D}$ with new state-action-reward transition pairs. Before that, DQNs fail to converge. After the memory buffer $\mathcal{D}$ is updated, agents are able to learn the optimal channel bonding policy for scenario-B.

Next, we conduct experiments on six WLANs with different sizes. Their main attributes are shown in Fig. 3.11, where (a) and (b) present the user satisfaction

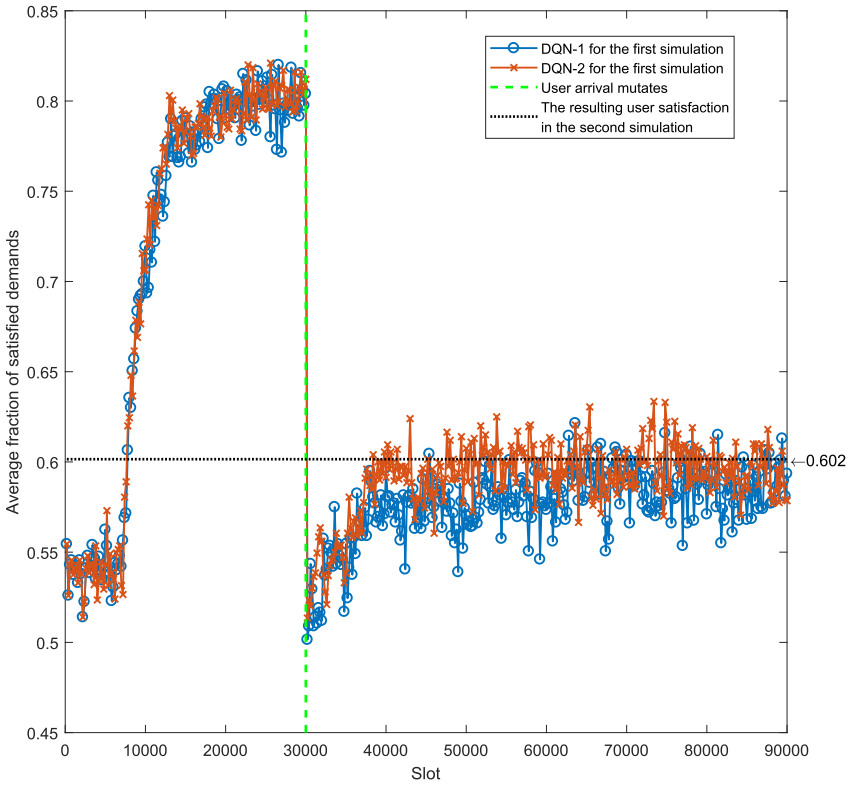

Figure 3.10: The impact of varying user arrival rates.

achieved by the tested algorithms for different WLANs. We can see that DQNs always have the highest user satisfaction under different scenarios, meaning that they are suitable for large-scale WLANs. For example, Fig. 3.11 (a) shows that when there are $N = 15$ APs, the average fraction of satisfied demands achieved by DQN algorithms is 0.88, which exceeds other algorithms by 35% to 120%. These results show that the increase in the number of APs N does not significantly impact the performance of DQN algorithms. It is worth noting that, when there are $N = 5$ APs, the average fraction of satisfied demands and percentage of satisfied users achieved by TRL is 0.85 and 0.96, respectively. These results are no different to DQN algorithms because the state space is small and TRL is able to converge to the optimal action. However, as the number of APs N increases to 30, TRL experiences a sharp decline in user satisfaction. After $N \geq 15$, the user satisfaction achieved by TRL is the same as Random. These results show that the performance of TRL algorithm is closely related to the number of APs N. This is because TRL is not suited for large state space. As the network scale increases, the state space will

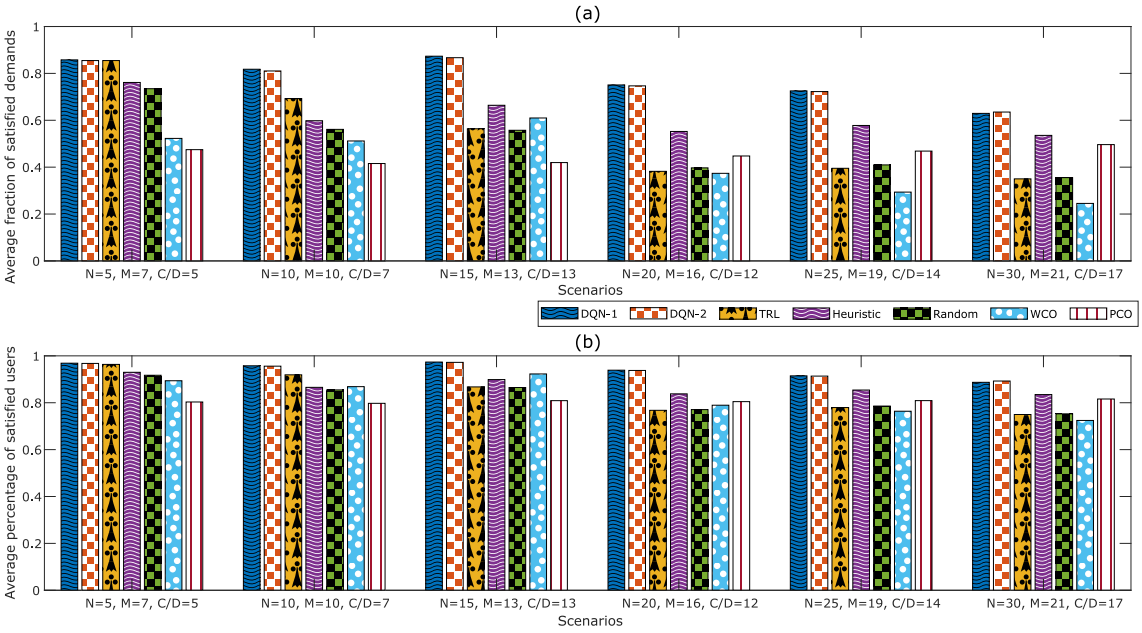

Figure 3.11: Performance of algorithms in different WLANs, where N and M are number of APs and number of channels, respectively. The term C/D denotes the maximum number of users that can be satisfied by a single channel.

increase exponentially. Therefore, TRL agents only visit a few states and thus most Q-values are not updated at all, meaning the actions taken by agents are arbitrary. Interestingly, after checking the Q-table of TRL agents, we find that when $N = 30$, 85.9% of Q-values are zero on average, meaning that TRL agents do not learn to bond channel. By contrast, when $N = 5$, only 53.3% of Q-values are zero. Therefore, we conclude that for large-scale WLANs, a neural network instead of Q-tables is necessary to ensure good performance.

This section also makes the following remarks. There are also experiments that study the impact of hyper-parameters such as the learning rate α, the reward decay rate γ and the mini-batch size N_{mb}. Experimental results show that these hyper-parameters do not affect the results significantly, where the average deviation on user satisfaction is only 3% when using different hyper-parameter values. Lastly, Jain's fairness index is also used to compare the user satisfaction obtained by ten APs. The proposed DQN approach has a Jain's fairness index of 0.92.

3.4 Conclusion

This chapter aims to equip APs with the ability to bond one or more channels under unknown and random traffic demands scenarios. It has proposed two DRL algorithms to maximize user satisfaction. Numerical results show that the proposed DRL algorithms are able to learn the optimal channel bonding policy that satisfies time-varying traffic demands and improves user satisfaction by up to 60% as compared with greedy and fixed channel bonding algorithms. In addition, the results also show that the proposed DRL algorithms have a much higher performance than tabular Q-learning; advantageously, they are adaptive to changing traffic conditions.

As mentioned in Chapter 1, a key concern is the EE of Wi-Fi networks. Also, future Wi-Fi networks may aim to satisfy user demands and also the energy requirement of IoT devices. Therefore, the next chapter proposes a DRL approach to meet both requirements.

Chapter 4

Learning to Charge RF-Energy Harvesting Devices in Wi-Fi Networks

This chapter addresses the problem of sustaining RF-energy harvesting devices or energy users that operate in a Wi-Fi network. As discussed in Chapter 2, prior works that study this problem, e.g., [10, 12, 37–39], do not consider APs with EH capability. By contrast, this chapter considers an EH AP with stochastic energy arrivals. This is important because EH APs can reduce carbon emissions and operating expenditure [9]. This chapter also addresses the bringing challenge due to random energy arrivals. That is, an AP needs to manage the use of energy to pervert from future energy outages. In addition, their solution has no learning ability and thus requires non-causal information, such as perfect Channel State Informatica (CSI) to RF-energy harvesting devices, and non-causal energy arrivals. As mentioned in Chapter 1, collecting the current CSI of devices will consume additional energy and incur significant delays to legacy data users.

To address these research gaps, this chapter proposes two machine learning-based solutions that only rely on historical CSI and energy arrivals. Note, this information is collected by an AP over time via the IEEE 802.11k standard which allows an AP

to evaluate its resources in a WLAN. Specifically, these two machine learning-based solutions aim to derive a transmit power level to satisfy both the data rate and energy requirement of users. The first solution, i.e., (Deep Q-network) DQN [22], determines the best transmit power for an AP given its energy level and legacy data user channel gain. The second solution, i.e., (Model Predictive Control) MPC [23], relies on Gaussian Process Regression (GPR) [24], a machine learning method, to predict an AP's future harvested energy and channel gains to legacy users. Lastly, both solutions are readily deployable in current APs.

The rest of this chapter is structured as follows. Section 4.1 presents a solar-powered Wi-Fi model with two types of users. The DQN and MPC solutions are outlined in Section 4.2 and their analysis is in Section 4.3. The conclusion of this chapter is in Section 4.4.

4.1 System Model and Problem

Time is discretized into T slots, and the set of time slots is $\mathcal{T} = \{1, 2, \ldots, T\}$. Each slot is one second in length; this means the terms power and energy can be used interchangeably. There are N IoT devices uniformly located around an AP. Let $\mathbb{D}$ be the set of IoT devices. Similarly, there are U legacy data users. In each time slot, the AP serves one data user.

The AP has a battery of size B_{max}. Its energy arrival is governed by the Markovian model presented in [80]. Specifically, the model contains four different solar states: 'Excellent', 'Good', 'Fair', and 'Poor'. Each state represents a different solar intensity throughout a day. In the j-th state, the energy arrival x (in mJ) is a random value drawn from a Normal distribution $\mathcal{N}(x \,|\, \mu_j, \sigma_j)$ with mean μ_j and variance σ_j. Then, the energy harvested by the AP in slot t is $\widetilde{E}_t = x_t \Phi \overline{\eta}$, where Φ and $\overline{\eta}$ is the panel size and the solar energy conversion efficiency. The energy level

of the AP, denoted as B_t, evolves as per,

$$B_t = \min(B_{max},\, B_{t-1} - P_{t-1} + \widetilde{E}_t), \tag{4.1}$$

where P_t (in mW) is the transmit power of the AP at time slot t, which is bounded by P_{max}. Note, the AP can be powered by other energy sources, e.g., winds, where the difference to the solar model is the distribution of energy arrivals.

Block fading is considered. The AP is aware of the CSI to *legacy* or data users but is unaware of the CSI to IoT devices. Let d_i denote the Euclidean distance from node i to the AP, and g_i^t is the channel gain between the AP and node i, and it is defined as,

$$g_i^t = \frac{1}{{d_i}^2}|Z|^2, \tag{4.2}$$

where Z is drawn from a complex normal distribution $\mathcal{CN}(\mu, \sigma^2)$.

IoT devices has a battery with capacity b_{max}. Let b_i^t be the current battery level of IoT device i. In each slot, IoT devices receive a charge whenever the AP transmits. The receive signal power p_i^t at IoT device i is calculated as $p_i^t = g_i^t P_t$. A practical 2.4 GHz non-linear RF-energy harvester [81] is considered. For IoT device i, the function $\beta(p_i^t)$ returns the RF-energy conversion rate given incident power p_i^t.

Each IoT device consumes a fixed amount of energy, denoted as $\hat{E}$ (in mJ), to sample and return its data to the AP. If its energy level satisfies $b_i^t < \hat{E}$, device i will not carry out any operation. Mathematically, b_i^t evolves as follows,

$$b_i^t = \begin{cases} b_i^{t-1} + p_i^t \beta(p_i^t), & b_i^t < \hat{E}, \\ \min\left(b_{max}, b_i^{t-1} + p_i^t \beta - \hat{E}\right), & \text{Otherwise.} \end{cases} \tag{4.3}$$

In each time slot, the AP transmits to a random data user $u \in U$ that has a channel gain of g_u^t. Let $\gamma_u^t = p_u^t / N_0$ be the Signal-to-Noise Ratio (SNR) of user u, where $p_u^t = g_u^t P_t$ is the received signal strength and N_0 is the white noise power. As

per the Shannon-Hartley formula, its theoretical maximum data rate is,

$$r_u^t = W \log_2(1 + \gamma_u^t), \tag{4.4}$$

where W is the bandwidth of the channel.

Without loss of generality, all users in U are assumed to have a fixed data rate requirement r_{min}. Define an indicator $J^t(P_t)$ that indicates whether the user u of time slot t is satisfied if the AP uses transmit power P_t. Formally, we have

$$J^t(P_t) = \begin{cases} 1 & \text{if } r_u^t \geq r_{min} \\ 0 & \text{Otherwise.} \end{cases} \tag{4.5}$$

Similarly, we have the following indicator that represents whether *all* IoT devices are able to collect a sample and transmit in slot t. Formally,

$$I^t(P_t) = \begin{cases} 1 & \text{if } b_i^t \geq \hat{E}, \forall i \in \mathbb{D} \\ 0 & \text{Otherwise.} \end{cases} \tag{4.6}$$

Define $S(.)$ as $S_t(P_t) = I^t(P_t)J^t(P_t)$. Then, the problem is as follows: find the optimal transmit power policy π^* that returns the transmit power level P_t for each time t that maximizes

$$Z_1 = \lim_{T\to\infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=1}^{T} S(P_t)\right]. \tag{4.7}$$

An alternative objective is to consider energy efficiency, which is given by

$$\eta_t(P_t) = \begin{cases} \dfrac{I^t(P_t)J^t(P_t)}{P_t}, & P_t \neq 0, \\ 0, & P_t = 0. \end{cases} \tag{4.8}$$

In words, if users meet their data rate requirement and all IoT devices are able to transmit, then Eq. (4.8) returns the non-zero value $1/P_t$. This means by improving energy efficiency, an AP will spend less energy to satisfy the requirement of both

data users and IoT devices.

In this regard, the second objective is to maximize the energy efficiency of the AP in order to support both types of users. Formally, the second objective is to maximize

$$Z_2 = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T} \eta_t(P_t)\right]. \tag{4.9}$$

4.2 Solutions

This section first proposes a DRL solution, i.e., Deep Q-network (DQN) [17], that allows an AP to learn the best transmit power by interacting with the system over time. Then, it presents another solution, i.e., Model Predictive Control (MPC) [23], which solves for the HAP's transmit power iteratively over a planning horizon. More specifically, it relies on Gaussian Process Regression (GPR) [24], which uses *historical* information to estimate future system states. The system states include the channel gains of data users and RF-energy harvesting users and energy arrivals.

4.2.1 Solution-1: Reinforcement Learning

This section first re-formulates the optimization problem in Section 4.1 as a Markov Decision Process (MDP) [16], which is then solved using a DQN [17]. In particular, there is an agent, e.g., AP, that observes the state of the system, and takes a corresponding action, which results in a reward.

4.2.1.1 MDP

Define a tuple with four elements $(S, A, P(s_{t+1}|s_t, a_t), R(s_{t+1}|s_t, a_t))$. The state space is S, where $s_t \in S$ represents the state at time t. The action space is A, where $a_t \in A$ is the action taken by the agent at t. The transition probability to state s_{t+1} after taking action a_t is defined as $P(s_{t+1}|s_t, a_t)$. Lastly, the function $R(s_{t+1}|s_t, a_t)$ returns the reward r_t after taking action a_t at state s_t. Let π be the policy taken by an agent, where $\pi(s_t)$ returns the action a_t for state s_t. The agent's goal is to find

the optimal policy π^* that maximizes the following expected discounted cumulative reward

$$\mathbb{E}\left[\sum_{t=0}^{+\infty}\gamma^t R(s_{t+1}|s_t,\pi(s_t))\right], \tag{4.10}$$

where $\gamma \in [0,1]$ is the discount factor, which scales the significance of future rewards.

Now, the optimization problem at hand is instantiated an MDP as follows. Let the state s_t be a tuple $s_t = (g_u^t, B_t)$ that includes the channel gain of a user and the battery level of the AP. The action a_t corresponds to the transmit power level $P_t \in [0, P_{max}]$. If the AP's battery B_t has insufficient energy to support the action a_t chosen by the agent, then the action is set to $a_t = B_t$.

The optimization problem at hand contains two different objectives Z_1 and Z_2, which correspond to (4.7) and (4.9), respectively. To this end, this section defines two reward definitions: **Reward-1** and **Reward-2**. For Reward-1, the reward r_t for state s_t and action a_t is defined as $r_t = I^t(P_t)J^t(P_t)$. In particular, if the AP takes an action that is able to satisfy both types of users, then the reward is one; otherwise, it is zero. The second reward definition, namely Reward-2, relates to the energy efficiency $\eta_t(.)$ achieved by the AP at slot t, and is defined as $r_t = \eta_t(.)$. That is, if the AP uses Reward-2, then its goal is to achieve higher energy efficiency. Note, unless stated explicitly, the proposed approach will apply Reward-1. The aforementioned transition of state-action-reward satisfies the Markov property. That is, the battery level in the next state s_{t+1} depends on the current state battery level and the action taken at slot t. Also, the historical battery level and channel gain do not impact the transition of the current state s_t to the next state s_{t+1}.

To ensure that the solution is practical, the transition probability $P(s_{t+1}|s_t, a_t)$ is assumed to be unknown, and hence the said MDP is model-free. This is because the distribution of CSI and energy arrivals is not readily available in practice. Moreover, it ensures that the solution is applicable in current Wi-Fi networks. To solve the formulated MDP, a reinforcement learning approach, namely, Deep Q-network (DQN) is applied to learn the optimal policy π^* that maximizes (4.7) or (4.9).

4.2.1.2 DQN

The aim of a DQN is to learn the state-action-value (also known as Q-value) over time [17]. Specifically, define Q-value $Q(s_t, a_t)$ as the expected discounted cumulative reward for action a_t taken at state s_t. Formally,

$$Q(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma^t R(s_{t+1}|s_t, a_t)\right]. \tag{4.11}$$

It is updated as per Bellman's equation,

$$Q'(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\hat{Q}(r_t, s_{t+1}, a_{t+1}) - Q(s_t, a_t)), \tag{4.12}$$

where

$$\hat{Q}(r_t, s_{t+1}, a_{t+1}) = r(s_t, a_t) + \gamma \max Q(s_{t+1}, a_{t+1}). \tag{4.13}$$

In (4.12) and (4.13), $Q(s_t, a_t)$ and $\hat{Q}(r_t, s_{t+1}, a_{t+1})$ are the evaluated and targeted Q-values, respectively. The term $Q'(s_t, a_t)$ is the updated Q-value. The variable $\alpha \in (0, 1]$ is the learning rate. Therefore, a DQN aims to minimize the average temporal difference-error of Q-values, defined as,

$$L(\theta) = \min \mathbb{E}[(\hat{Q}(r_t, s_{t+1}, a_{t+1}, \theta') - Q(s_t, a_t, \theta))^2]. \tag{4.14}$$

Upon convergence, the parameter θ can be used to retrieve any state-action value.

A DQN uses two neural networks to estimate and store Q-values. The first neural (evaluate) network, which is denoted θ, is used to evaluate Q-values $Q(s_t, a_t)$, and the second neural (target) network, denoted as θ', outputs targeted Q-values $\hat{Q}(r_t, s_{t+1}$. The training data is sourced from a memory buffer that stores historical state-action-reward pairs. For every K slots, the DQN is trained using the Stochastic Gradient Descent method [22] with a learning rate α. For every K' slots, where $K' \gg K$, θ' is replaced by θ.

Fig. 4.1 shows how a DQN agent or AP selects an action. At time slot t, the

AP observes the environment (state s_t). Then, the evaluation network θ outputs the evaluated Q-value $Q(s_t, a_t)$ for each action. An agent then selects an action using the ϵ-greedy policy, where it executes the action with the maximum Q-value with probability $(1 - \epsilon_t)$. Otherwise, a random action is taken. To ensure sufficient exploration of the action space, the agent diminishes ϵ_t as per,

$$\epsilon_t = \max \left[1, \epsilon_T + \frac{\epsilon_{inc}}{(t+1)^2} \right], \tag{4.15}$$

where ϵ_0 and ϵ_T are the initial and final exploration rate, respectively. The term ϵ_{inc} is the diminishing rate of ϵ_t. The agent then observes the resulting reward r_t, and uses it to update its evaluation network θ according to Eq. 4.14.

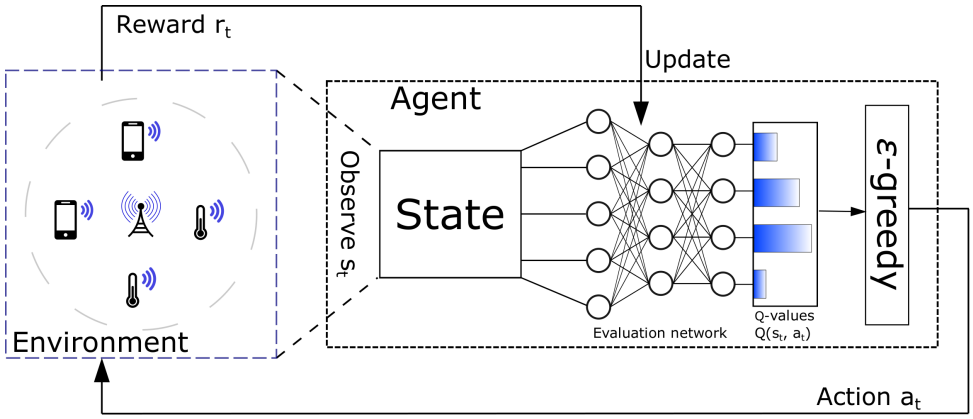

Figure 4.1: An illustration of the DQN approach that is run by an AP.

4.2.2 Solution-2: MPC

MPC [23] is used to choose control actions over a time horizon. It relies on a prediction model to estimate system dynamics, e.g., prices, weather, heating requirements. The prediction is then used to build a virtual system model that simulates the changes of a real system several slots ahead. Formally, let the system state be x_τ. Denote the control action u_τ taken in state x_τ as $v_\tau(x_\tau, u_\tau)$. Let $\mathbf{u}_t = [u_t, u_{t+1}, \ldots, u_{t+L}]$ be a vector of L control actions. Define $\mathbf{A}$ as the collection of possible actions over L time slots. The average performance over L time slots is

thus

$$V_t(\mathbf{u}) = \frac{1}{L+1} \sum_{\tau=t}^{t+L} v_\tau(x_\tau, u_\tau). \tag{4.16}$$

For each time slot t, the goal is determine the optimal control action $\mathbf{u}$ to maximize a performance index V_t over the time window $[t, t+L]$. Mathematically,

$$\mathbf{u}^* = \arg\max_{\mathbf{u}\in\mathbf{A}} V_t(\mathbf{u}). \tag{4.17}$$

The corresponding optimal average performance is denoted as V_t^*. Given $\mathbf{u}^*$, the action u_t^* is then executed in time slot t. After that (4.17) is then solved for time slot $t+1$ and so forth.

In our case, the system state x_t consists of the solar energy arrival at the AP, the channel gain to data user s, namely g_s^t, and lastly, we have the last recorded channel gain value of IoT devices, i.e., $\mathbf{g}^{t-1} = \{g_i^{t-1} \mid i = 1, \ldots, N\}$. Formally, we have, $x_t = (\widetilde{E}_t, g_s^t, \mathbf{g}^{t-1})$. Note, the CSI of IoT devices in the previous slot is readily available because IoT devices can return their sensing data and their CSI to the AP at the same time. Also, if an IoT device fails to return its data due to energy shortage, it will return its CSI in the next transmission around. The control action u_t is the transmit power with range from $[0, P_{max}]$. In particular, we have $\mathbf{A} = [0, p_{max}]^L$; the set of transmit power used over L time slots. The performance of a given state and control is defined as $v_\tau(x_t, u_t) = I^t(u_t)J^t(u_t)$ for objective Z_1. For the second objective, it is defined as $v_\tau(x_t, u_t) = \eta_t(u_t)$; see Eq. (4.9).

MPC employs GPR [24] to predict the system state from t to $t+L$. A GPR model $\mathcal{G}$ can be trained as a probabilistic non-parametric black-box to identify a non-linear dynamic system. Given a set of training data $\{(q_i, p_i) \mid i = 1, 2, \ldots\}$, a GPR model learns the predictive response value p' for a new input value q'. A GPR model is trained using the following linear regression model,

$$p = q^T w_{\mathcal{G}} + \xi_{\mathcal{G}}, \tag{4.18}$$

where the coefficient $w_{\mathcal{G}}$ and the error variance $\hat{\sigma}^2$ of $\xi_{\mathcal{G}} \sim \mathcal{N}(0, \hat{\sigma}^2)$ are learned from the training data. In our case, let q_i be the i-th slot and p_i be a component of the system state x_t. There is a separate GPR model for each network parameter. As an example, consider the GPR model for energy arrivals at the AP. Define set $\mathcal{D}_t$ with size K to store the energy arrivals in the last K slots; formally, we have $\{(i, \widetilde{E}_i); i = t-K, t-K-1, \ldots, t\}$. In each slot, the GPR model is trained using $\mathcal{D}_t$, and is then used to predict future energy arrival.

Algorithm 1 illustrates the MPC solution. Line-1 initializes $\mathcal{D}_t$, where it gathers K slots worth of data in order to train the GPR model of each component of the system state. At line-6, the function *Update*$(x_t, \mathcal{D}_{t-1})$ (i) adds the system state x_t into $\mathcal{D}_{t-1}$, (ii) deletes the oldest data x_{t-K} from $\mathcal{D}_{t-1}$, and (iii) returns a new $\mathcal{D}_t$. After that, $\mathcal{G}$ is called to update the GPR of each component of the system state $\mathcal{D}_t$ by calling the function *Train*(.) in line-7. The function *Predict(.)* then returns L predictions; namely, $\{x_{\tau+1}, \ldots, x_{\tau+L}\}$. Given the predicted system states, in line-9, MPC calls *Optimize(.)* to determine the optimal transmit power $\mathbf{u}$ that maximizes (4.16). After that, the optimal transmit power is applied for time slot t, and is then recorded in the vector $\mathbf{u}^*$.

Algorithm 1: The proposed MPC algorithm.

Input: K, L, T
Output: $\mathbf{u}^*$

1 $\mathcal{D}_0 = \text{CollectData}()$;
2 $\mathcal{G}_0 = \text{Train}(\mathcal{D}_0)$;
3 $\mathbf{u}^* = \emptyset$;
4 **for** $t = 1, 2, \ldots, T$ **do**
5 $\quad x_t = \text{GetSystemState}()$;
6 $\quad \mathcal{D}_t = \text{Update}(x_t, \mathcal{D}_{t-1})$;
7 $\quad \mathcal{G}_t = \text{Train}(\mathcal{D}_t)$;
8 $\quad [x_{\tau+1}, \ldots, x_{\tau+L}] = \text{Predict}(\mathcal{G}_t, \mathcal{D}_t)$;
9 $\quad u_t^* = \text{Optimize}(x_{\tau+1}, \ldots, x_{\tau+L})$;
10 $\quad \text{Transmit}(u_t^*)$;
11 $\quad \mathbf{u}^* \cup u_t^*$;
12 **end**
13 return $\mathbf{u}^*$;

4.3 Evaluation

Simulations are conducted using Python 3.5 with TensorFlow 1.0 and Scikit-learn 0.21, and MATLAB 2017. Experiment parameters correspond to an actual Wi-Fi network with parameter value(s) set as per Table 4.1. For example, data users are uniformly distributed within a range of five meters to 25 meters from the AP. RF-energy devices are uniformly distributed within 11 meters from an AP. Note, this range ensures IoT devices receive some energy. Beyond 11 meters, the reward is zero because one or more IoT devices are unable to receive sufficient energy. The solar energy conversion efficiency $\overline{\eta}$ is 15% [82]. The energy requirement $\hat{E}$ per sample for an RF-energy device is 1.38 mJ, which includes circuit energy consumption [83]. Also, the action space is equally discretized to N_A actions. This is because Q-learning cannot handle a continuous action space; otherwise, it would result in infinite output neurons. The interval between two adjacent actions is P_{max}/N_A Watts. The DQN agent and MPC controller are installed at the AP; hence, the agent is not limited by computational power. Also, through experimentation, DQN contains four fully-connected layers with 500 neurons in order to accurately model Q-values. Lastly, the GPR uses the Radial Basis Function (RBF) as the kernel to predict system states [24].

DQN and MPC are compared against the following algorithms/solutions/rules:

- **Tabular RL (TRL)**: The state, action and reward are the same as that of DQN. Similarly, TRL uses the same learning rate α and reward decay rate γ as DQN's. However, the difference is that TRL uses a table to store Q-values and does not have the memory replay feature. The state space is discretized due to the fact that TRL is not able to handle continuous state space. The discretization level is 10^3 which means each element of the state contains 10^3 integer values.

- **Greedy**: In every slot, the AP uses P_{max} to transmit; otherwise, it uses a transmit power level that consumes all the energy in its battery.

Table 4.1: Parameter values used in experiments.

Model Parameters	***Value***
Number of IoT Devices N	5
The minimum user distance d_{min}	5 meters
The maximum user distance d_{max}	25 meters
The maximum device distance $\hat{d}_{max}$	11 meters
The maximum battery of the AP B_{max}	100 J
The maximum battery of IoT devices b_{max}	50 mJ
Solar panel size Φ	15 cm^2
Solar energy conversion efficiency $\overline{\eta}$	15% [82]
The maximum AP transmit power P_{max}	200 mW
The mean of channel gain μ	1
The variance of channel gain σ^2	0.1
Bandwidth of the channel W	20 MHz
Energy requirement per sample $\hat{E}$	1.38 mJ [83]
User data rate requirement r_{min}	133 Mbps
Total simulated time slots T	150,000
White noise power N_0	10^{-6} W

Algorithm Parameters	***Value***
Learning rate α	10^{-5}
Reward decay rate γ	0.4
Number of actions N_A	100
Memory size of DQN	50,000
Mini-batch size N_{mb}	200
Time interval to update θ K	Every two slots
Time interval to update θ' K'	Every 400 slots
Replay start time slot	3,000-th
Structure of DQN	Four fully connected layers
Number of neurons	500
Activation function of neurons	Leaky ReLU
Initial exploration rate ϵ_0	1
Final exploration rate ϵ_T	0.01
Testing start slot	120,000-th
Size of $\mathcal{D}_t$	20
GPR kernel	RBF
Rolling/receding horizon L	4

- **Random**: The AP uniformly chooses a transmit power level in the range $[0, P_{max}]$.

- **No-policy**: The AP is not aware of IoT devices and aims to only meet the requirement of legacy data users. For data user u, the transmit power P_t is calculated as

$$P_t = \frac{N_0(2^{\frac{r_{min}}{W}} - 1)}{g_u^t}. \tag{4.19}$$

It is worth noting that the above mentioned competing rules are derived from practical setups of a WiFi network. They return a specific action for an AP in each slot. This action will then be taken by an AP which will observe the resulting reward.

Define 3,000 slots as an episode. For each episode, the average value of the following metrics is calculated and recorded:

- **Number of activated IoT devices** per slot, which is calculated as $\frac{1}{3000}\sum_{t+3000}^{t} n_t$, where n_t. This metric corresponds to the average number of IoT devices that achieve sampling per slot in one episode.

- **Fraction of satisfied data users**. This is calculated as $\frac{1}{3000}\sum_{t+3000}^{t} I^t(.)$, meaning the number of satisfied data users over one episode of slots.

- **Energy efficiency**, which is equal to $\frac{1}{3000}\sum_{t+3000}^{t} \eta_t(.)$. This metric is the average energy efficiency achieved by an AP in one episode.

- **Reward**. This is defined as the average reward gained by the agent within one episode; it is calculated as $\frac{1}{3000}\sum_{t+3000}^{t} r_t$, where r_t refers to Reward-1.

The following experiment investigates the energy efficiency $\eta_t(.)$ of the AP assuming it has no energy constraint; e.g., when it is connected to the grid. The agent applies Reward-2, namely energy efficiency. The experimental results are presented in Fig. 4.2 and 4.3. From Fig. 4.2, we see that MPC achieves the highest energy efficiency of around 6.5. In terms of DQN and TRL, we notice that energy efficiency increases because the RL agent learns to determine the optimal transmit power over

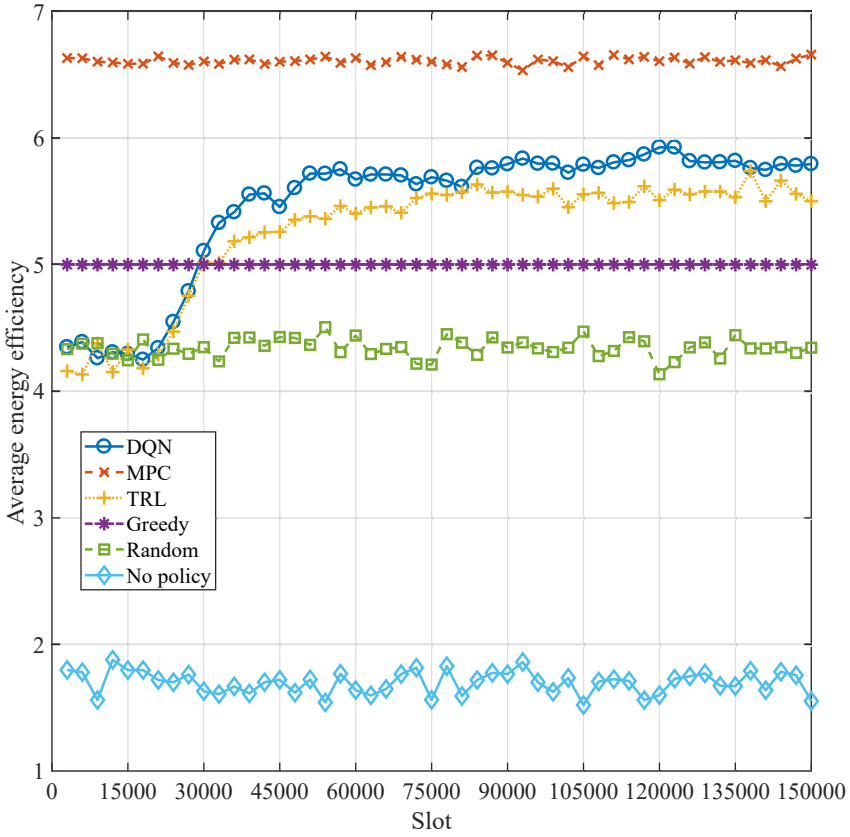

Figure 4.2: Elapsed time versus energy efficiency.

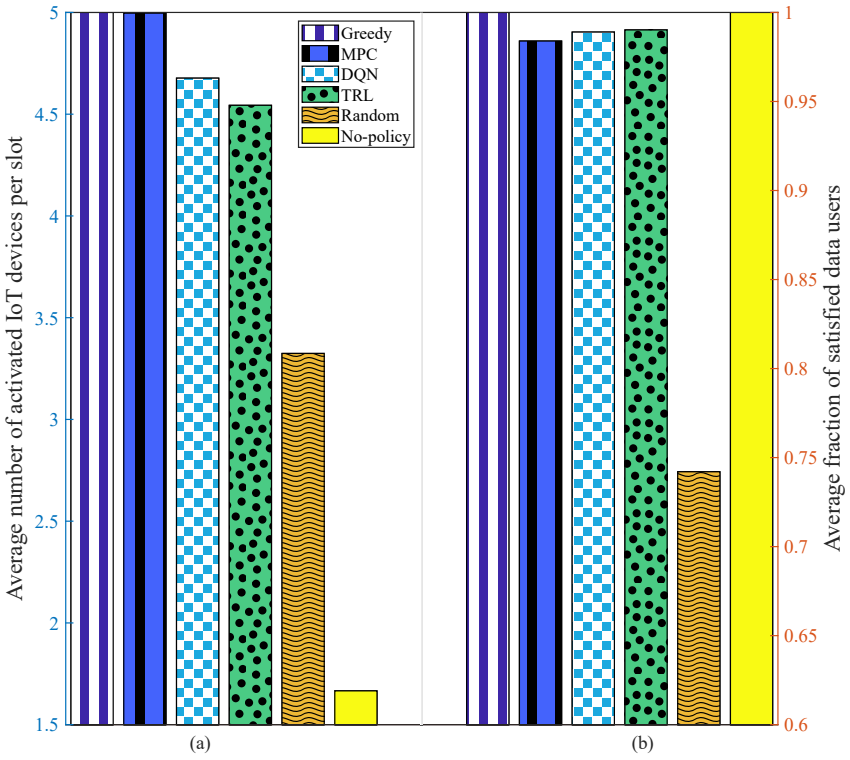

Figure 4.3: The satisfaction of both types of users. (a) Average number of activated IoT devices per slot. (b) Average fraction of satisfied data users.

time. Before the 15,000-th slot, the agent has no knowledge of the environment. Hence, as we can see from Fig. 4.2, the energy efficiency achieved by both DQN and TRL agent is only 4.4. After training, DQN converges at around the 50,000-th slot. We can see that DQN gains the second-best energy efficiency among the tested algorithms, with around 5.8 on average. By contrast, TRL improves energy efficiency to only 5.5. Also, we see that the energy efficiency of TRL is worse than DQN. This is because DQN takes advantage of neural networks that allow it to deal with continuous and large state space. In our problem, the CSI is random and continuous values, so the state space is large. Also, the memory replay strategy used by DQN breaks the correlation between adjacent states, which helps speed up convergence.

For non-RL approaches, such as Greedy, Random and No-policy, the energy efficiency shown in Fig. 4.2 remains roughly the same. For example, the energy efficiency achieved by Random is always 4.4. This is because the transmit power used by Random is uniformly distributed in the range $[0, 0.2]$ mW, meaning that the average energy consumption per slot/transmit power is 0.1 mW. Such a transmit power level is not sufficient to meet the energy/data rate demands of users. In terms of No-policy, the energy efficiency is around 1.8 because the transmit power used by No-policy is not sufficient to activate every IoT device. As we can see from Fig. 4.3, the number of activated IoT devices is only 1.7

Fig. 4.3 demonstrates the average value of two important metrics, where the left y-axis is the number of activated IoT devices per slot and the right y-axis is the fraction of satisfied data users. From Fig. 4.3, we see that Greedy has the highest satisfaction for both IoT and legacy data users, with five active devices per slot with a user satisfaction of 100%, respectively. This is because Greedy is able to use the maximum transmit power (200 mW) because there is no energy limitation in this scenario. MPC also achieves almost 100% user satisfaction with five active devices per slot. In terms of data users, MPC achieves a satisfaction value of 0.98. However, from Fig. 4.2, we see that energy efficiency achieved by Greedy is always 5.0, which is lower than MPC by 35%. This means MPC uses 35% less energy to

achieve almost the same performance as Greedy. As for DQN, Fig. 4.3 shows that the average number of activated devices per slot is 4.65 and the satisfaction of data users is 0.98. This indicates that MPC performs better than DQN in terms of the number of supported IoT devices where it supports 7% more IoT devices than DQN. Fig. 4.2 and 4.3 indicate that compared to Greedy, DQN supports 6% less number of IoT devices per slot but its energy efficiency is 16% higher than Greedy. Also, we see that MPC outperforms DQN by around 10% in terms of energy efficiency. Compared to DQN, MPC uses not only the current CSI of legacy data users but also the historical CSI of RF-energy devices. Consequently, the GPR predictor is able to generate accurate system state predictions and thus allows MPC to choose the optimal action. Lastly, we see that No-policy gains 100% user satisfaction but it only supports 1.7 IoT devices per slot on average. These results confirm that the proposed solutions, i.e., DQN and MPC, are able to effectively charge RF-energy IoT devices.

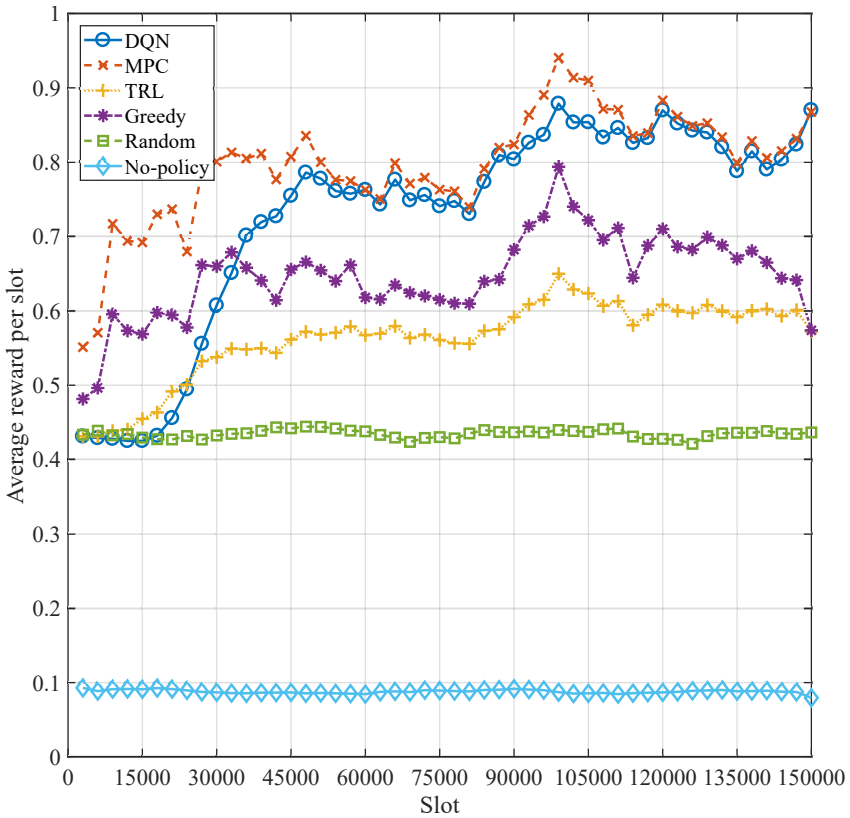

Figure 4.4: Elapsed time versus reward gained by the tested algorithms/schemes.

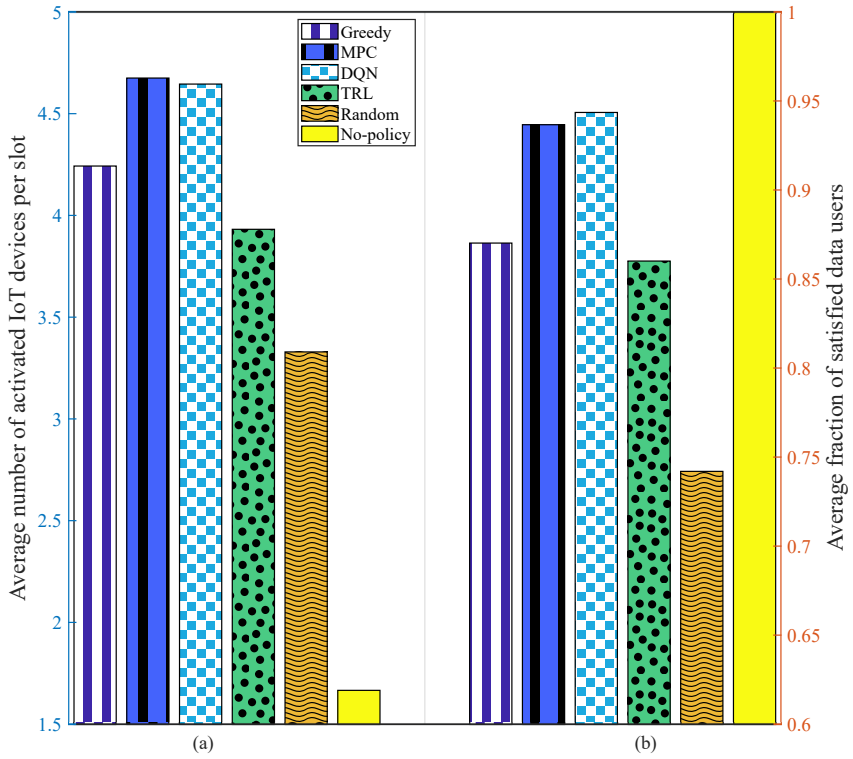

Figure 4.5: The satisfaction of both types of users under a random energy arrival scenario. (a) Average number of activated IoT devices per slot, and (b) Average fraction of satisfied data users.

Next, we consider an imperfect energy supply. The solar panel size Φ is set to 15 cm^2. The energy arrival at the AP is random. Fig. 4.4 shows the reward gained by different algorithm/approaches over 150,000 time slots. Fig. 4.5 shows the resulting satisfaction of both type of users. From Fig. 4.4, we notice that the reward increases significantly because the RL agent learns to use energy over time. Before the 15,000-th slot, the average reward gained by both the DQN and TRL agent is only 0.43. After training, the DQN agent gains around 0.8 rewards on average, while TRL improves the reward to only 0.6. Also, we see from Fig. 4.4 that DQN gains around 20% to 30% more reward than TRL. As a result, Fig. 4.5 shows that DQN is able to support 20% more IoT devices than TRL, where the number of activated IoT devices achieved by DQN and TRL is 4.61 and 3.9, respectively.

In terms of non-RL algorithms, Fig. 4.4 shows that MPC gains a reward between 0.75 and 0.9 which is the same as the well-trained DQN agent after the 60,000-th slot. This indicates that both MPC and DQN have converged and they apply the

policy with the maximum reward. In addition, MPC supports 4.65 IoT devices and 0.94 data users per slot on average. Combining the results in Fig. 4.2, we notice that the performance of MPC declines as compared to DQN. The reason is that in this experiment, user satisfaction depends on not only channel gains but also energy arrivals. This significantly narrows down the difference between the user satisfaction achieved by DQN and MPC. However, we can still notice that at the 97,500-th slot, MPC gains around 5% more reward than DQN. We also see that Greedy gains around 0.65 reward on average. However, its performance is worst than the case when the AP has no energy limitation. This is because it does not conserve energy, meaning it causes energy outages. As for the Random rule, from Fig. 4.4, it only gains 0.44 reward on average. This is because Random only uses 0.1 mW energy per slot, leading to battery overflow in around 90% time slots. No-policy gains only 0.1 reward; see Fig. 4.4. The reason is that the number of activated IoT devices achieved by No-policy is only 1.7 per slot on average, meaning IoT devices require more slots to harvest energy until they have sufficient energy to gather a sample. It is worth noting that the performance of DQN and MPC is not significantly affected by random energy arrivals. Comparing Fig. 4.3 and 4.5, when the AP has imperfect information of its energy arrivals, the user satisfaction of DQN and MPC reduces by around 8% to 4% for both types of users. By contrast, we see from Fig. 4.3 and 4.5 that the performance achieved by TRL, Greedy and Random drops significantly when we consider random energy arrivals. For example, the average number of activated IoT devices achieved by TRL is 3.9 per slot, whereas it is 4.55 when the AP is powered by a perfect energy source. Also, we see that in Fig. 4.4, TRL takes about 25,000 more time slots to reach convergence than that in Fig. 4.2. The reason is that the state space must also include different energy levels at the AP and CSI of users, so the state space is larger. In the case of random energy arrivals, the shortcomings of TRL when for large state space are evident. Moreover, results show that when we consider an imperfect energy source, transmit power control is critical. The reason is that the AP may encounter energy shortages, which reduces system

performance. This is evident in Fig. 4.3 and 4.5, the user satisfaction of Greedy and TRL reduces by around 11% to 15%.

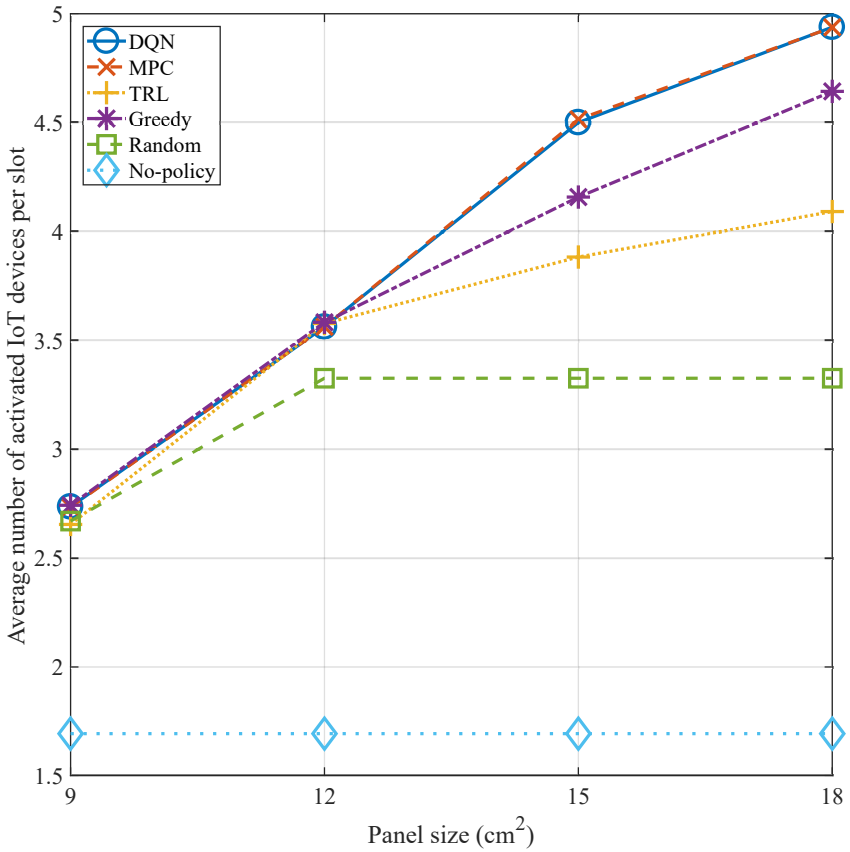

Figure 4.6: Varying solar panel sizes versus the number of activated devices.

Fig. 4.6 and 4.7 illustrate how different solar panel sizes Φ impact user satisfaction. From Fig. 4.6 and 4.7, we see that the satisfaction of both types of users increases as the panel size increases. The average number of activated IoT devices of DQN and MPC increases from 2.75 to 4.9, with a 90% increment as the panel size doubles. Also, the performance of data users increases by 40%, from 0.71 to 0.98. This is because the AP harvests more energy when using a larger solar panel. This allows them to use a higher transmit power to meet the needs of those users that are far away from the AP. In terms of Greedy and TRL, both of them are able to support 3.55 IoT devices per slot on average, and achieve 80% satisfaction for data users when the solar panel size Φ is 12 cm^2, see Fig. 4.6 and 4.7. However, they are inferior to DQN and MPC when Φ is larger than 12 cm^2. As shown in Fig. 4.6 and 4.7, the number of activated IoT devices achieved by Random is always 3.3 per

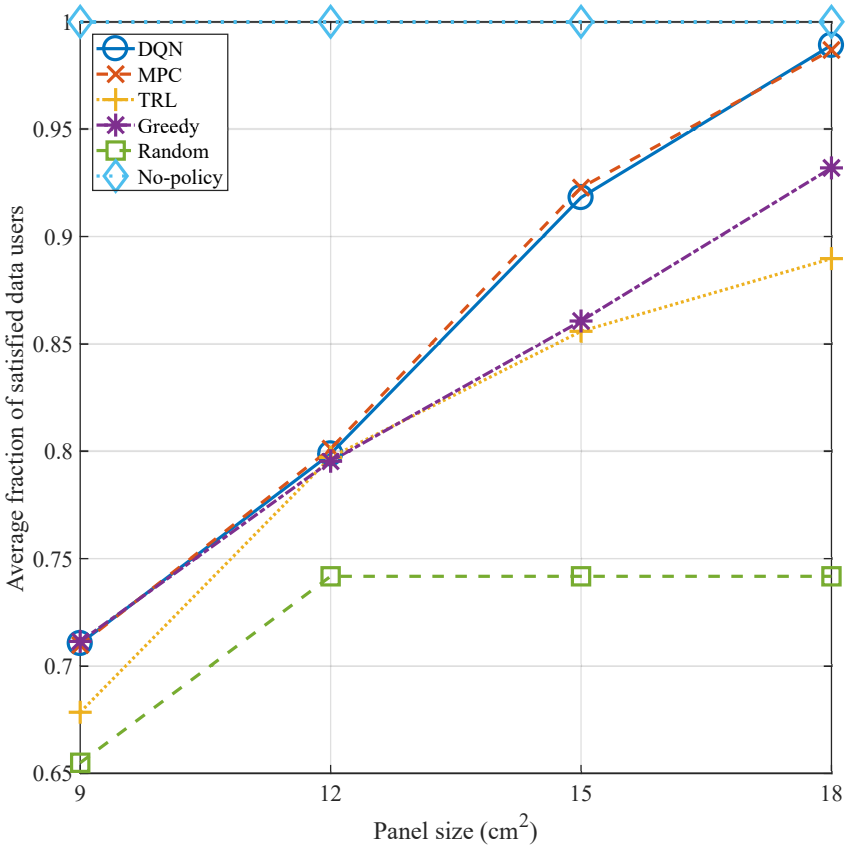

Figure 4.7: Varying solar panel sizes versus the fraction of satisfied data users.

slot when the solar panel size is larger than 12 cm^2. This is because its average energy consumption is 0.1 mW. Moreover, as the solar panel size increases, higher energy arrivals lead to a higher overflow rate. As shown in Fig. 4.6 and 4.7, the satisfaction of both data users and IoT devices gained by No-policy also remains unchanged as the solar panel size Φ increases. This indicates that the overflow rate of No-policy is higher than other solutions. The number of activated IoT devices is only 1.7. The reason is that the AP is not aware of IoT devices and thus those devices cannot harvest enough energy. Fig. 4.6 and 4.7 also show that the difference in user satisfaction achieved by different algorithms/schemes is wider as the panel size increases. This means energy management is more necessary when the solar energy arrival rate is large. This is because when the panel size is smaller than 12 cm^2, the AP is not able to harvest sufficient energy to meet the data/energy requirements of users. By contrast, DQN and MPC are able to learn the optimal transmit power to maximize user satisfaction. Therefore, we see that DQN and MPC are significantly

superior to the other tested algorithms when the panel size is larger than 15 cm^2

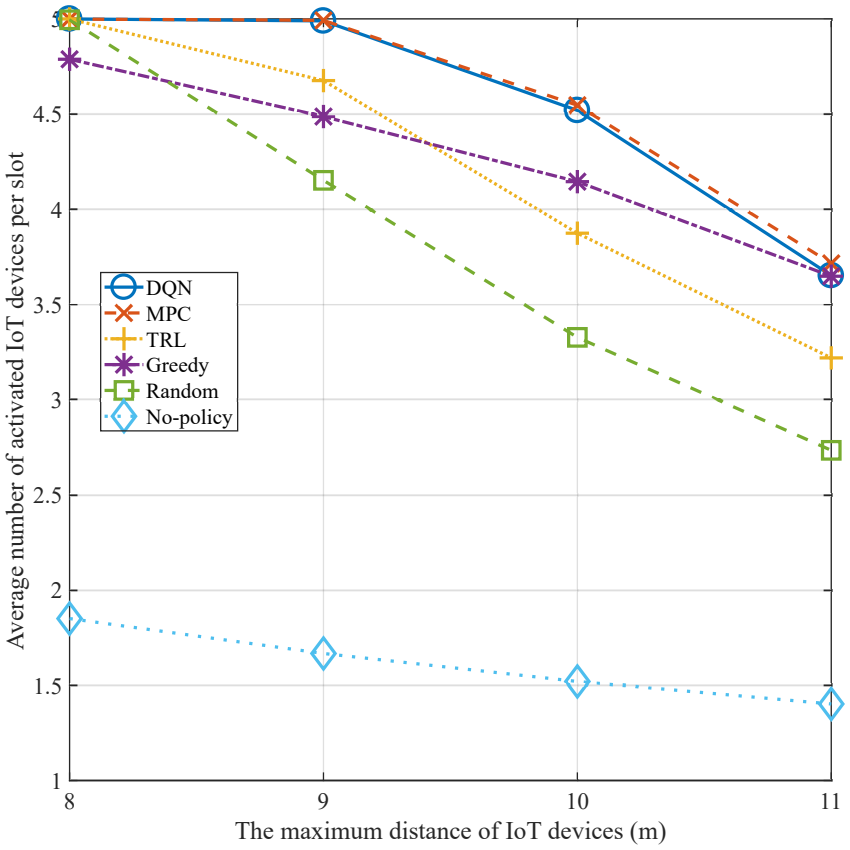

Figure 4.8: Average device distance versus the number of activated devices.

We now study different cell sizes. Fig. 4.8 and 4.9 show that the satisfaction of both types of users decreases by 35% to 45% as the distance/cell size increases. This is because the channel gain becomes smaller if the user distance increases. However, we see that DQN has minimal degradation in user satisfaction as the cell size becomes larger. For example, the fraction of satisfied data users decreases from 0.98 to 0.87, an 11% drop as the IoT device distance increases from 20 to 26 meters. As shown in Fig. 4.9, DQN is able to improve the percentage of satisfied data users by 8% to 10% when the maximum user distance is under 24 meters. Fig. 4.8 shows that MPC achieves the same number of activated IoT devices to DQN as the maximum IoT device distance increases from eight to ten meters. However, referring to Fig. 4.9, MPC is no better than DQN in terms of the fraction of satisfied data users, which is the second-best algorithm among the tested algorithms. Fig. 4.9 also shows that as the cell size increases, No-policy supports fewer RF-energy IoT devices. It only

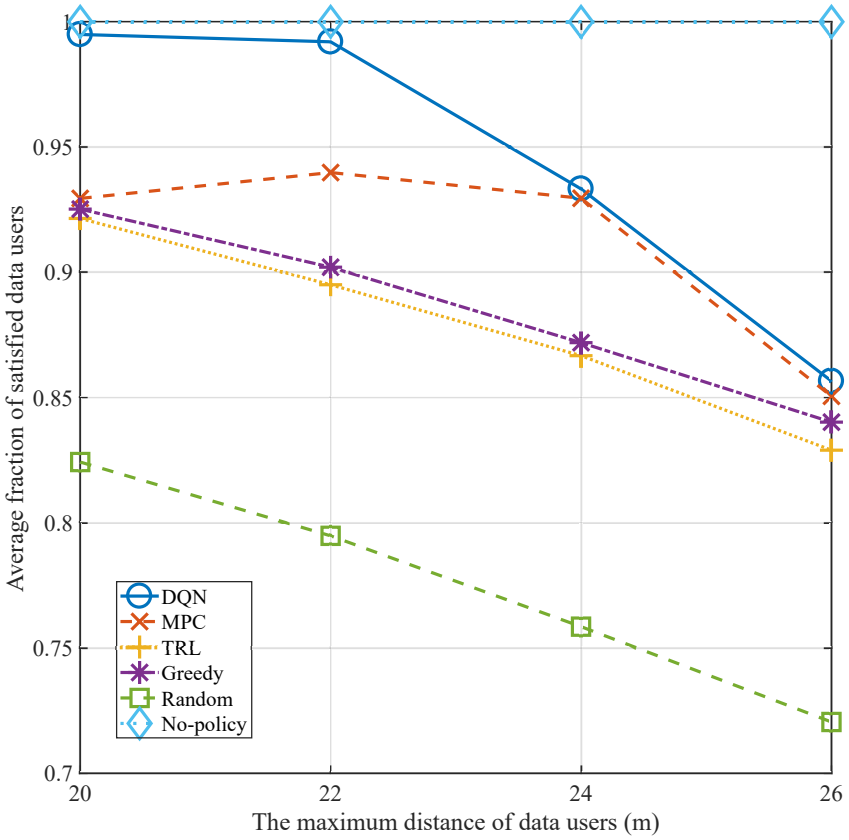

Figure 4.9: Maximum user distance versus the fraction of satisfied data users.

supports 1.4 devices per slot when the maximum IoT device distance is 11 meters. However, the user satisfaction of data users achieved by No-policy is unchanged; see Fig. 4.9. The reason is that No-policy is only aware of data users so it will not increase transmit power if there is an energy shortage at IoT devices.

We see from Fig. 4.8 and 4.9 that the user satisfaction achieved by different tested algorithms converges to a smaller value as user distance increases. For example, in Fig. 4.9, the difference in the fraction of satisfied data users achieved by DQN and MPC converges to 0.01 as the maximum distance of data users increases to 26 meters. This reveals that the AP cannot gain high user satisfaction through transmit power management as the distance increases. The reason is that the channel gain becomes smaller with increasing user/device distance. Consequently, the received power at devices/users is too small for data receptions or energy harvesting no matter what transmit power policy is adopted by the AP.

4.4 Conclusion

This chapter has shown how an AP can learn to adapt its transmit power when serving data users and simultaneously ensure IoT devices receive sufficient energy. The work in this chapter is significant because existing Wi-Fi networks will likely be used to support RF-energy harvesting IoT devices. According to the results, the proposed DQN and MPC algorithms power 10% to 42% more IoT devices and gain 9% to 27% more user satisfaction as compared to competing algorithms. Also, the results show that the proposed algorithms are able to determine the optimal transmit power for devices and users according to the current and historical battery status of an AP.

As discussed in Chapter 1, channel bonding is an alternative approach to improve the EE of an AP. This is because channel bonding potentially increases data rate, and thus increases the EE of an AP. To this end, the next chapter will employ both transmit power control and channel bonding to improve the EE experienced by an AP.

Chapter 5

Energy-Efficient Channel Bonding and Transmit Power Control

As discussed in Chapter 2, existing works on channel bonding, such as [15, 29–31, 34, 84, 85], have not considered Energy Efficiency (EE) optimization. They aim to maximize throughput, guarantee fair capacity to APs, or assign bandwidth to APs corresponding to their traffic load. In addition, past works that consider transmit power control, e.g., [43, 46–48], do not assign a bonded channel to APs.

Lastly, past works require cooperation between APs [15, 84], where they require global information such as traffic load at APs and their assigned channels [19, 32, 34, 86]. However, this chapter proposes a decentralized solution, where an AP only has local information. Specifically, an AP is only aware of its traffic load and channel state information reported by its associated users. The traffic load of neighboring APs and the interference received by associated users is unknown to an AP.

To this end, this chapter addresses a novel problem whereby an AP has to independently learn the optimal channel bonding and transmit power allocation policy in order to maximize its EE as well as minimize queuing delays and overflows. More specifically, it needs to balance the trade-offs between interference, energy consump-

tion, throughput, and queue length. Ideally, an AP should use a channel with no interference and offering sufficient capacity for its traffic. However, the issue is that the amount of interference from its neighboring APs is random, which is governed by random channel gains and traffic arrivals. In addition, an AP must not maximize EE at the expense of long queues or overflows. This chapter first uses a novel six-state Continuous Time Markov Chain (CTMC) to study this problem and obtain some insights for the EE optimization problem over different traffic load scenarios. Next, the channel and transmit power allocation problem is formulated as a Markov Decision Process (MDP) [16], which is then solved using a Dueling Deep Q-Network (DDQN) [25] run by an AP. The proposed DDQN solution allows an AP to adapt to time-varying traffic load and to minimize its queuing delay, with no knowledge of traffic arrivals and the interference strength from neighboring APs.

Section 5.1 formalizes a Wi-Fi model. Section 5.2 presents the said CTMC and presents numerical results for different traffic scenarios. After that, Section 5.3 outlines the MDP and discusses the DDQN solution. Section 5.4 presents simulation results, and Section 5.5 concludes this chapter.

5.1 System Model and Problem

Fig. 5.1 shows a WLAN with an AP i and its associated user u. The user experiences inter-cell interference from APs outside the cell. Without loss of generality, assume the interferer is a neighboring AP, indexed by j. Time is divided into T slots; each slot has duration $\tau = 1$ second, and thus the term power and energy are interchangeable. There are N channels; each channel has a bandwidth of M MHz. Let $\mathcal{C} = \{C_1, \ldots, C_N\}$ denote the set of channels. Each AP is allocated a so called *primary* channel, and it is able to bond one or more *secondary* channels. Let ζ_i^t be the set of channels assigned to AP i at slot t. There are a total of $|\zeta_i^t|$ channels. The bandwidth B_i^t of AP i is calculated as $B_i^t = M \cdot |\zeta_i^t|$. Also, let ζ_j^t as the channel used by the interfering AP j.

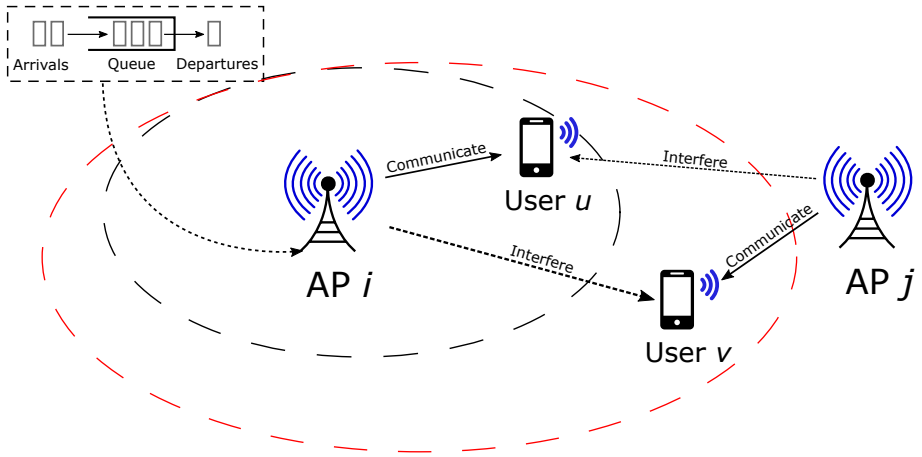

Figure 5.1: An example WLAN. Each AP maintains a data queue. As indicated by the red and black ellipses, AP i can use different transmit power levels, which result in different EE and data rates to user u. User u may experience interference when AP j is transmitting data to its user v on the same channel.

Define I_t as a binary indicator that returns one if AP i and interferer j are on the same channel; otherwise, it returns zero. Formally, we have,

$$I_t = \begin{cases} 0, & \zeta_i^t \cap \zeta_j^t = \emptyset, \\ 1, & \text{Otherwise.} \end{cases} \tag{5.1}$$

Define $g_{x,y}^t$ to be the channel gain from an AP or interferer x to its user y at slot t. Block fading is considered, where $g_{x,y}^t$ is a constant within one slot but it varies slot by slot independently. Assume the path loss $PL(d_{x,y})$ (in dB) between x and y follows the Log-distance path loss model [87], expressed as

$$PL(d_{x,y}) = PL(d_0) + 10\,\overline{\gamma}\log_{10}\left(\frac{d_{x,y}}{d_0}\right) + \mathcal{X} \tag{5.2}$$

where $d_{x,y}$ is the distance between AP x and user y, $PL(d_0)$ is the path loss at the reference distance d_0 from AP x, $\overline{\gamma}$ is the path loss exponent, and $\mathcal{X}$ (in dB) is drawn from a normal distribution given by $\mathcal{N}(0, \sigma^2)$.

Let p_x^t be the transmit power of an AP j, where $p_x^t \leq P_{max}$ Watts. Then, given

Table 5.1: Key Notations

Notation	***Description***
$i,\ j$	AP i and j
$u,\ v$	User u and v
t	The t-th slot
T	The total number of time slots
K	The total number of channels
M	The bandwidth of one channel
$\mathcal{C}$	Set of channels
ζ_i^t	The channels assigned to AP i
B_i^t	The bandwidth assigned to AP i
I_t	Interference indicator
$g_{x,y}^t$	The channel gain from AP x to user y
p_t^i	The transmit power of AP i
N_0	The white noise power per Hertz
q_i^t	The total bits queuing at AP i
L	The maximum queue length
d_i^t	The number of transmitted bits in slot t
γ_u^t	The SINR of user u
$\hat{p}_i^t$	The power consumption of AP i
η_i^t	Energy efficiency
π	Channel and transmit power policy
λ_i	The arrival rate of user requests
μ_i	The departure rate of user requests
s_k	The k-th state of the CTMC
P_c	The probability of bonding channels
Π_k	The steady-state probability of state s_k

noise power N_0, the Signal to Interference plus Noise Ratio (SINR) of user u is

$$\gamma_u^t = \frac{p_i^t \cdot g_{i,u}^t}{N_0 \cdot B_i^t + I_t \cdot p_j^t \cdot g_{j,u}^t}, \tag{5.3}$$

where $g_{i,u}^t$ and $g_{j,u}^t$ are respectively the channel gain from AP i and AP j to user u. The symbol p_i^t and p_j^t denote the transmit power used by AP i and j, respectively. The symbol B_i^t denotes the bandwidth of AP i. As per the Shannon-Hartley formula, the theoretical data rate $\hat{r}_i^t$ (in bits/s) from AP i to user u is given by

$$\hat{r}_i^t = B_i^t \log_2(1 + \gamma_u^t). \tag{5.4}$$

Each AP has a finite data queue that stores up to L bits. At time t, the total queued bits at an AP x is q_x^t, and the number of arrived bits is a_x^t. We assume the data arrival a_x^t is governed by a Poisson process with an arrival rate λ_x. As shown later in Section 5.4, the data arrival a_x^t can be drawn from an actual traffic tracefile such as [1]. The total number of transmitted bits d_x^t within time slot t is

$$d_x^t = \min(\hat{r}_x^t, q_x^t). \tag{5.5}$$

The data queue for AP x evolves as per

$$q_x^{t+1} = \min(L, q_x^t + a_x^t - d_x^t). \tag{5.6}$$

We define $\hat{p}_i^t$ (in mW) as the total power consumption of AP i. We adopt the power consumption model presented in [3] which models the Radio Frequency (RF) chain of an IEEE 802.11ac Network Interface Card (NIC). Specifically, it models the energy consumption of the Digital-to-Analog Converter (DAC), power amplifier, baseband processing, mixers, antennas and filters on an IEEE 802.11ac NIC. The power consumption $\hat{p}_i^t$ (in collection of policiesmW) is a linear function of bandwidth B_i^t, transmit power p_t^i and number of transmitted bits d_i^t of AP i, which is given by,

$$\begin{aligned} \hat{p}_i^t = B_i^t[\Phi_1 N_1 + \Phi_2 N_2 \log_2 B_i^t + f(N_2)] \\ + \Phi_3 N_1 + \Phi_4 d_i^t + \Phi_5 p_t^i + \overline{P}, \end{aligned} \tag{5.7}$$

where N_1 and N_2 correspond to the number of antennas and spatial streams at an AP, respectively, $\overline{P}$ is an AP's base power consumption, $\Phi_1, \ldots, \Phi_5$ and f are model parameters. In Eq. (5.7), the energy consumed by the DAC, modulator, and Inverse Fast Fourier Transform (IFFT) is respectively given by the term $\Phi_1 N_1$, $\Phi_2 N_2 \log_2 B_i^t$ and $f(N_2)$, which are proportional to the channel bandwidth.

The EE, denoted as η_i^t, of AP i is defined as the amount of data (in bits) transmitted per Joule of energy consumed by the AP. The unit of EE is bits per Joule.

Formally, The EE of AP i is calculated as

$$\eta_i^t = \begin{cases} d_i^t/\hat{p}_i^t, & p_t^i \neq 0, \\ 0, & p_t^i = 0. \end{cases} \tag{5.8}$$

This means an AP with higher EE can use less energy to transmit per bit of data. Note, the EE of an AP is considered only when it transmits (busy) and assumes it consumes zero energy or has zero EE when it is idle. However, in practice, as per [3], an AP will have a base power consumption when it does not transmit (idle). This *fixed* base power consumption can be included in results without changing the conclusions/findings as it only scales results by a fixed constant.

Assume the interfering AP uses only its primary channel and does not bond channels. As per (5.3)-(5.8), the EE of AP i depends on the channel ζ_i^t and the transmit power p_t^i assigned to AP i. Let Ω be a collection of policies, where a policy $\pi \in \Omega$ governs the transmit power level p_t^i and channel allocation ζ_i^t at each time t of AP i. Note, we have $0 \leq p_t^i \leq P_{max}$ and $\zeta_i^t \subseteq \mathcal{C}$.

Define the following objective function:

$$Z(\pi) = \lim_{T\to\infty} \mathbb{E}^{\pi} \left[\sum_{t=1}^{T} \beta_1 \eta_i^t - \beta_2 q_i^t \right]. \tag{5.9}$$

where β_1, $\beta_2 \in (0, 1)$ are weights that balance EE and queue length, and $\mathbb{E}^{\pi}[.]$ denotes the expectation when using policy π. Note that the reason for including the queue length in 5.9 is that there is a risk of queue overflow if an AP only considers optimizing its EE. Indeed, as noted by [88], there is a trade-off between EE and throughput. That is, increasing the EE of a network may decrease its throughput, which may further lead to queuing delays and overflows. The aim is to find the optimal policy in Ω, denoted as π^*, that yields the maximum long-term expected EE and minimum queue length. Mathematically, we have

$$\pi^* = \arg\max_{\pi\in\Omega} Z(\pi). \tag{5.10}$$

5.2 Analysis

To gain some insights into the problem at hand, this section models the channel bonding process in a WLAN with two APs, denoted as i and j, as a CTMC. There are two users, denoted as u and v; they are associated to AP i and j, respectively. Without loss of generality, assume there are two adjacent channels C_1 and C_2. Assume that AP i is allocated C_1 as its primary channel, and it is able to bond the secondary channel C_2. Assume that AP j functions as the interfering AP and it is assigned C_2 as its primary channel. This means if AP i transmits using the bonded channel C_1 and C_2 while AP j is busy, both APs will experience interference.

Fig. 5.2 shows the CTMC under consideration. User requests at AP i are modeled as a Poisson process with an arrival rate of λ_i per second. AP i uses a channel to serve user u for a random amount of time that is modeled as a negative-exponential distribution with mean $1/\mu_i$ second. The service rate μ_i (in requests per second) of AP i is modeled as a Poisson process. Similarly, user request arrivals and departures for AP j are modeled as an independent Poisson Process with parameters λ_j and μ_j, respectively. An AP is said to be *busy* if it receives a request from its user and then accesses a channel.

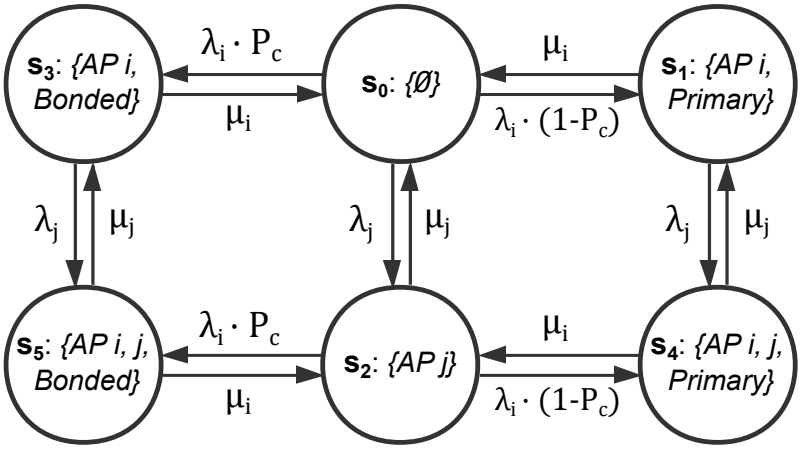

Figure 5.2: The transition diagram of the CTMC with six states.

Let $\mathcal{S} = \{s_0, \ldots, s_5\}$ be the set of CTMC states. A state s_k, where $k \in \{0, 1, \cdots, 5\}$, represents i) whether AP i and j are busy, and ii) if AP i is busy,

whether it is using a bonded channel. Table 5.2 shows the meaning of each CTMC state.

Table 5.2: The network states represented by the CTMC.

State	**AP** i	**AP** j	**Channel of AP** i	**Interference**
s_0	*Idle*	*Idle*	*Primary*	✗
s_1	*Busy*	*Idle*	*Primary*	✗
s_2	*Idle*	*Busy*	*Primary*	✗
s_3	*Busy*	*Idle*	*Bonded*	✗
s_4	*Busy*	*Busy*	*Primary*	✗
s_5	*Busy*	*Busy*	*Bonded*	✓

The transitions between states are defined as follows. The initial state is s_0. If AP i receives a request, it will use either a single channel with a probability $(1 - P_c)$ or a bonded channel with a probability P_c, which determines whether the CTMC moves to state s_1 or s_3, respectively. Similarly, if AP j receives a request in state s_0, the CTMC will move to state s_2. If an AP finishes its service before another AP receives a request, the CTMC will move back to state s_0. Otherwise, the CTMC will move to state s_4 or s_5, where both APs are busy. For example, if AP j receives a request when the state is s_1 or s_3, the CTMC will move to state s_4 or s_5. Note, AP i is assumed to not change its channel before it finishes a user request. Therefore, the CTMC cannot directly move from state s_3 to s_4 or from state s_1 to s_5, and vice-versa. In addition, since user request arrivals and departures are independent and time-continuous processes, a request arrival to an AP and a request departure from another AP do not happen at the same time. Hence, there is no direct transition from state s_1 and s_3 to s_2 or s_4 and s_5 to s_0, and vice-versa.

Define Π_k as the steady-state probability of state s_k. We can obtain the steady-state probability vector $\mathbf{\Pi} = \{\Pi_0, \Pi_1, \ldots, \Pi_5\}$ by solving the following balance equa-

tions,

$$(\lambda_i + \lambda_j)\Pi_0 = \mu_i\Pi_1 + \mu_j\Pi_2 + \mu_i\Pi_3, \tag{5.11}$$

$$(\mu_i + \lambda_j)\Pi_1 = \lambda_i(1 - P_c)\Pi_0 + \mu_j\Pi_4, \tag{5.12}$$

$$(\mu_j + \lambda_i)\Pi_2 = \lambda_j\Pi_0 + \mu_i\Pi_4 + \mu_i\Pi_5, \tag{5.13}$$

$$(\mu_i + \lambda_j)\Pi_3 = \lambda_i P_c\Pi_0 + \mu_j\Pi_5, \tag{5.14}$$

$$(\mu_j + \mu_i)\Pi_4 = \lambda_j\Pi_1 + \lambda_i(1 - P_c)\Pi_2, \tag{5.15}$$

$$(\mu_i + \mu_j)\Pi_5 = \lambda_i P_c\Pi_2 + \lambda_j\Pi_3, \tag{5.16}$$

$$\sum_{k=0}^{5} \Pi_k = 1. \tag{5.17}$$

The steady-state probability of each state can be shown to be

$$\Pi_0 = \Gamma\mu_i\mu_j, \tag{5.18}$$

$$\Pi_1 = \Gamma(1 - P_c)\lambda_i\mu_j, \tag{5.19}$$

$$\Pi_2 = \Gamma\lambda_j\mu_i, \tag{5.20}$$

$$\Pi_3 = \Gamma P_c\lambda_i\mu_j, \tag{5.21}$$

$$\Pi_4 = \Gamma(1 - P_c)\lambda_i\lambda_j, \tag{5.22}$$

$$\Pi_5 = \Gamma P_c\lambda_i\lambda_j, \tag{5.23}$$

where

$$\Gamma = (\lambda_i\lambda_j + \lambda_i\mu_j + \lambda_j\mu_i + \mu_i\mu_j)^{-1}. \tag{5.24}$$

The detailed derivation can be found in Appendix A.

Let $D(s_k)$ be the data rate of AP i when it is in state s_k. Let $P(s_k)$ be the power consumption of AP i in state s_k, which is calculated as per Eq. 5.7. According to Eq. 5.8, the EE experienced by AP i in state s_k is then calculated as $\eta(s_k) = D(s_k)/P(s_k)$. The steady-state probability Π_k can be treated as the expected long-term fraction of the time that the CTMC stays in state s_k [89]. Therefore, given

$T \to \infty$ slots, the average number of slots in which the CTMC stays in state s_k is given by $T\Pi_k$. Then, the average EE $\overline{\eta_i}$ experienced by AP i over T time slots is calculated as follows,

$$\overline{\eta_i} = \lim_{T\to\infty} \left[\frac{T\Pi_1\eta(s_1) + T\Pi_3\eta(s_3) + T\Pi_4\eta(s_4) + T\Pi_5\eta(s_5)}{T} \right], \tag{5.25}$$

$$= \Pi_1\eta(s_1) + \Pi_3\eta(s_3) + \Pi_4\eta(s_4) + \Pi_5\eta(s_5). \tag{5.26}$$

Note, AP i is idle in state s_0 and s_2, meaning it has zero EE; these states can be omitted from Eq. 5.25.

Table 5.3: Parameter values used in analysis.

Model Parameters	***Value***
Distance from a user to its associated AP	10 meters
Distance from a user to its interfering AP	10 meters
Operating frequency	5 GHz
Channel bandwidth M	20 MHz
Number of channels M	2
Noise power spectral density N_0	10^{-15} Watts/Hz
Transmit power of APs	1 Watts [90]
Distance to the reference point d_0	1 meter [91]
Number of antennas N_1	1 [3]
Number of spatial streams N_2	1 [3]
Variance of channel gain σ^2	0
Path loss exponent $\overline{\gamma}$	2 [91]
Request arrival rate λ_i and λ_j	2 [92]
Request departure rate μ_i and μ_j	2 [92]

Using the formulated CMTC, we now study different arrival rates λ, channel bonding probability P_c values and transmit power p_t^i. Unless stated explicitly, the parameter values listed in Table 5.3 will be applied. For instance, the distance from a user to its associated AP and to the interfering AP is 10 meters. In addition, as per the IEEE 802.11ac standard, the maximum transmit power of an AP is 30 dBm (one Watt) [90]. The worst case is when AP j always transmit at 1 Watt, meaning AP i will receive the maximum interference. At this point, assume no small-scale fading, meaning σ^2 is zero; small-scale fading will be discussed in Section 5.4.

Table 5.4: Parameter values of the power model in [3]

β_1	β_1	β_1	β_1	β_1	$f(N_2 = 1)$	$\overline{P}$
0.022	0.038	802.2	0.001	4.352	1.623	472.1

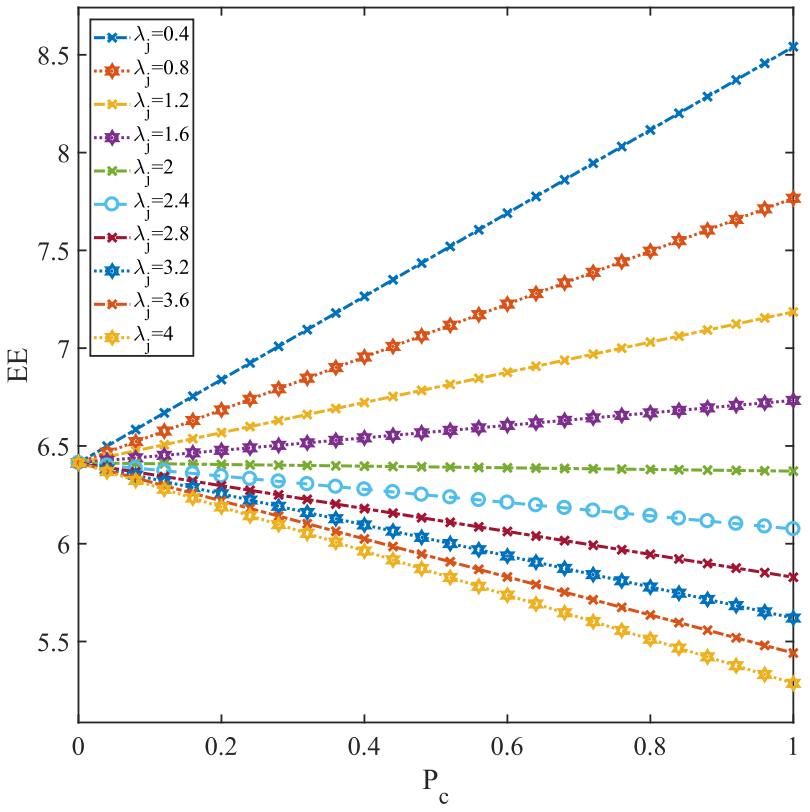

Figure 5.3: Different λ_j and P_c values.

This section shows the impact on EE for different user request arrival rates λ_j at AP j and channel bonding probability P_c values of AP i. From Fig. 5.3, we see that the EE of the WLAN depends on both λ_j and P_c. There is an inflection point located at around $\lambda_j = 2$ to determine whether bonding channels is beneficial to an AP. When the neighbor's traffic λ_j is smaller than 1.948, AP i will experience higher EE with increasing P_c value to bond channel more aggressively. The highest EE is achieved at 8.54 when $\lambda_j = 0.4$ and $P_c = 1$. However, when λ_j is larger than 1.948, using a bonded channel will reduce EE instead. The lowest EE is at 5.29, where $\lambda_j = 4$ and $P_c = 1$. Generally, AP i will experience higher EE if the traffic λ_j at the neighboring AP j is smaller. The reason is that a higher request arrival rate λ_j means that AP j will access channel C_1 more frequently. Hence, when bonding

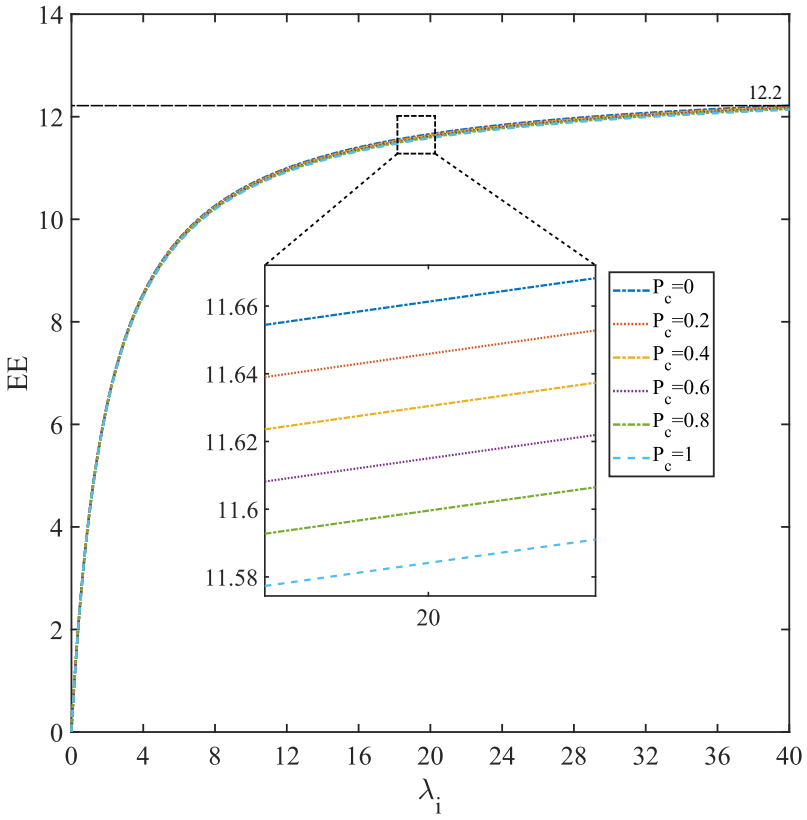

Figure 5.4: Different λ_i and P_c values.

channel C_0 and C_1, AP i will cause more interference. This reduces the capacity and EE of an AP.

We now study different traffic intensity λ_i at AP i and channel bonding probability P_c values. Fig. 5.4 shows that the EE is strongly related to λ_i but is not sensitive to P_c when λ_j is fixed. In general, the EE increases from 0 to 12.2 as λ_i increases from 0 to 40. One reason is that as λ_i increases, AP i holds the channel for a longer time, and thus transmits more bits in each slot. As a fixed circuit consumption at APs is considered, a higher data rate yields higher EE. Another reason is that as the arrival rate λ_i of AP i increases, the frequency of transitions to state s_1, s_3, s_4 and s_5 increases. This increases the steady-state probability Π_1, Π_3, Π_4, Π_5. Thus, according to Eq. 5.25, EE increases. Also, we can see that the EE experienced by AP j increases faster in $\lambda_i < 8$ than it is in $\lambda_i > 8$. The EE converges to around 12.2. This is because AP i will reach its maximum data rate as λ_i increases and thus the EE cannot be improved without a limit. Moreover, referring to the zoomed-in

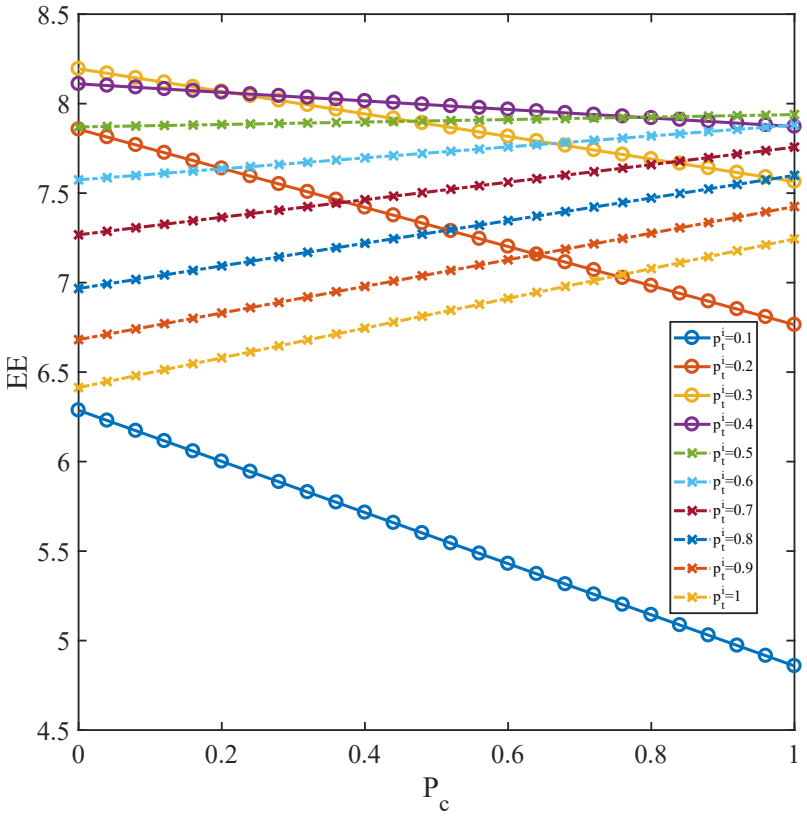

Figure 5.5: Different p_t^i and P_c values.

area in Fig. 5.4, at around $\lambda_i = 20$, we can see a 0.7% difference between the highest and lowest EE when using different P_c values. This shows that varying the value of P_c does not change the EE experienced by AP i under different traffic λ_i at AP i.

Fig. 5.5 compares different transmit power p_t^i and channel bonding probability P_c values. The distance between user u to AP j is 18 meters; at this distance, both p_t^i and P_c have a non-negligible impact on the EE experienced by AP i. Otherwise, EE will either increase or decrease monotonically with increasing P_c values. We see from Fig. 5.5 that when p_t^i is less than or equal to 0.4, as indicated by the solid lines, EE declines as P_c increases. In addition, EE is more sensitive to changing P_c for smaller transmit power p_t^i. For example, when $p_t^i = 0.1$, EE decreases from 6.29 to 4.86, which is the lowest EE value. We can also see the highest EE value is 8.2 by using $p_t^i = 0.3$ and $P_c = 0$. In contrast, if p_t^i is greater than 0.4, as indicated by the dash-dotted lines, EE increases as P_c increases. For example, when using a transmit power level of 1 W, EE increases from 6.41 to 7.24; this is an increase of 12%. In

addition, referring to Fig. 5.5, there is an inflection point located at $0.4 \leq p_t^i \leq 0.5$. That is, if p_t^i is less than 0.4, EE will increase as p_t^i grows. Otherwise, the growth in p_t^i will result in a lower EE value.

In conclusion, the EE of AP i varies with its transmit power level p_t^i and channel bonding probability P_c. More specifically, given a P_c value, the EE of AP i is determined by the traffic intensity λ_j of AP j rather than the traffic intensity λ_i of AP i. Also, the optimal transmit power level p_t^i of AP i depends on the channel bonding probability P_c value of AP i. Moreover, this section shows that there is an optimal channel bonding probability P_c value and transmit power level p_t^i for different scenarios. Hence, a solution needs to adapt both quantities depending on traffic load. This is the topic of the next section.

5.3 Solution

This section first formulates the previous problem as a Markov Decision Process (MDP) [16], which is then solved by a deep reinforcement learning solution; namely, Dueling Deep Q-Network (DDQN) [25]. The solution is aware of network conditions, i.e. queue length, the channel gain to users, and assigns the optimal channel(s) and transmit power level to an AP.

5.3.1 MDP

An MDP is formalized as a tuple $(\mathcal{Y}, \mathcal{A}, P(y_{t+1}|y_t, v_t), R(y_{t+1}|y_t, v_t))$ [16]. The state space is $\mathcal{Y}$, where $y_t \in \mathcal{Y}$ is the state at time t. The action space is $\mathcal{A}$, where $v_t \in \mathcal{A}$ is the state at time t. As illustrated in Fig. 5.6, there is an agent, e.g., AP, that observes the state of its environment, and takes an action. Then, the environment moves to a new state y_{t+1} with probability $P(y_{t+1}|y_t, v_t)$, and the AP earns a reward r_t which is given by the reward function $R(y_{t+1}|y_t, v_t)$. Let π be the strategy adopted by an agent, where $\pi(y_t)$ returns the action v_t for state y_t. The target of the agent

is to find the optimal strategy π^* that maximizes the following cumulative reward

$$\mathbb{E}\left[\sum_{t=0}^{+\infty}\gamma^t R(y_{t+1}|y_t,\pi(y_t))\right], \tag{5.27}$$

where $\gamma \in [0,1]$ is the discount factor for future rewards.

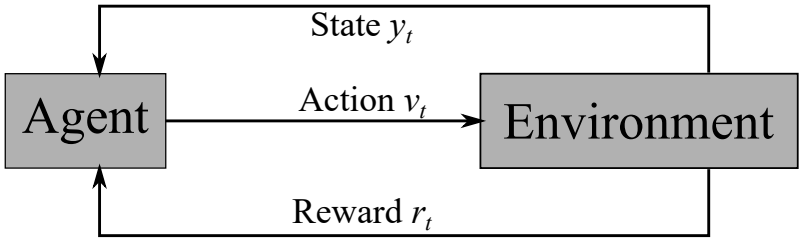

Figure 5.6: Interaction between an agent (AP) and its environment (Wi-Fi network).

Now, define the state, action, and reward as follows. **State**: Each state y_t includes the queue length q_i^t, channel gain to the associated user $g_{i,u}^t$, average arrival bits $\overline{a}_i^t$ and an indicator h_i^t. The indicator $h_i^t \in \{0,1\}$ returns one if the queue length reduces in slot t; otherwise, it returns zero. Mathematically, it is defined as

$$h_i^t = \begin{cases} 1 & \text{if } q_i^t < q_i^{t-1} \\ 0 & \text{Otherwise.} \end{cases} \tag{5.28}$$

The symbol $\overline{a}_i^t$ is the average number of arriving bits in the last ten slots, which is calculated as,

$$\overline{a}_i^t = \frac{1}{10}\sum_{t}^{t-10} a_i^t \tag{5.29}$$

Now, the state can be represented by the tuple $y_t = (q_i^t, g_u^t, \overline{a}_i^t, h_i^t)$. Note, the said quantities, i.e., traffic load and channel state information, can be obtained via the IEEE 802.11k standard, which provides an AP with the ability to measure interference experienced by users.

Action: The action v_t corresponds to the allocated transmit power level p_t^i and channel(s) ζ_i^t at AP i in slot t. Define a binary indicator b_i^t to determine the channel used by AP i. To be specific, if b_i^t returns one, AP i will use a bonded channel in slot

t, or use its primary channel otherwise. Then, the action is given by the following tuple

$$v_t = (p_t^i, b_i^t), \tag{5.30}$$

where $p_t^i \in [0, P_{max}]$ and $b_i^t \in \{0, 1\}$. As the action space is continuous, the transmit power p_t^i needs to be discretized into N_p levels. That is, from zero power to P_{max}, the interval between two adjacent transmit power levels is P_{max}/N_p Watts. Hence, the total number of actions is $|\mathcal{A}| = 2N_p$.

Reward: both EE and queuing delays are considered. To balance the trade-off between EE and queuing delays, a delay cost that is a function of the queue length at an AP will be introduced. This is important because of the following two issues. First, as discussed in Section 5.2, an AP is able to increase EE by reducing its transmit power in some cases. Second, as discussed in Section 5.2, if an AP uses a bonded channel more frequently, it may experience more interference from neighboring APs. These two issues reduce the data rate of APs, increase the queue length of an AP, and thus may result in queue overflows.

Hence, the reward is defined as

$$r_t = \begin{cases} -0.5 & \text{if } q_i^t + a_i^{t+1} > L \\ max(0, \beta_1 \cdot \eta_i^t - \beta_2 \cdot q_i^t) & \text{Otherwise.} \end{cases} \tag{5.31}$$

where the coefficients β_1, $\beta_2 \in (0, 1)$ aim to balance EE and queue length. The term $\beta_2 \cdot q_i^t$ is the delay cost. Also, the reward r_t is set to -0.5 in order to punish an AP for taking the actions that lead to a queue overflow.

Note, the Markov property for the aforementioned transition of state-action-reward holds. This is because the queue length q_i^{t+1} and the indicator y_i^{t+1} in the next state y_{t+1} only depend on the current queue length q_i^{t+1}, channel gain $g_{i,u}^t$ and the corresponding action v_t taken at slot t.

To ensure the solution is practical, the agent is assumed to have no prior knowledge of its environment, meaning the transition probability $P(y_{t+1} \mid y_t, v_t)$ is un-

known. To this end, a model-free reinforcement learning approach, namely, DDQN [25] will be applied at an AP to learn the optimal policy π^*.

5.3.2 Q-learning

This section first introduces the concept of Q-learning, a value-based RL solution [18]. The general steps of Q-learning are illustrated in Algorithm-2. The objective of Q-learning is to learn the optimal state-action-values (also known as Q-values) over time. To be specific, the Q-value function $Q^{\pi}(y_t, v_t)$ estimates the expected discounted cumulative reward for action v_t taken at state y_t under policy π. Mathematically, we have,

$$Q^{\pi}(y_t, v_t) = \mathbb{E}\left[\sum_{t=0}^{+\infty} \gamma R(y_{t+1}|y_t, v_t = \pi(y_t))\right]. \tag{5.32}$$

Intuitively, Q-values measure the *quality* of actions under a certain state. Q-values $Q^{\pi}(y_t, v_t)$ are updated according to the well-known Bellman equation,

$$Q^{\pi}(y_t, v_t) \leftarrow Q^{\pi}(y_t, v_t) + \alpha[\hat{Q}^{\pi}(r_t, y_{t+1}, v_{t+1}) - Q^{\pi}(y_t, v_t)], \tag{5.33}$$

with

$$\hat{Q}^{\pi}(r_t, y_{t+1}, v_{t+1}) = r_t + \gamma \max Q^{\pi}(y_{t+1}, v_{t+1}), \tag{5.34}$$

where $Q^{\pi}(,.,)$ and $\hat{Q}^{\pi}(,.,.,)$ is the evaluated and targeted Q-values, respectively. The hyper-parameter α is the learning rate and $\alpha \in (0, 1]$. The convergence of $Q^{\pi}(.) \rightarrow Q^{\pi^*}(.)$ is guaranteed as t tends to infinity [18]. After convergence, an agent or AP executes the optimal policy π^* by choosing the action that has the maximum Q-value in each state.

To avoid local optima and speed up convergence, the ϵ-greedy strategy is applied when choosing an action [18]. That is, an agent takes a random action with probability ϵ_t. Let ϵ_t be the exploration rate at time slot t. It diminishes over time as

per

$$\epsilon_t = \max\left[1, \epsilon_T + \frac{\epsilon_{inc}}{(t+1)^2}\right], \tag{5.35}$$

where ϵ_0 and ϵ_T are the initial and final exploration rate, respectively. The term ϵ_{inc} is the diminishing rate of ϵ_t.

Algorithm 2: Q-learning [18].

```
Initialize Q-values Q^π(y_t, v_t) = 0, ∀y_t ∈ 𝒴, ∀v_t ∈ 𝒜;
for t = 1, 2, ..., T do
    Observe y_t;
    Choose v_t according to the ϵ-greedy strategy;
    Take v_t, observe r_t and observe y_{t+1};
    Update Q^π(y_t, v_t) as per Eq. (5.33);
    Observe y_t ← y_{t+1};
end
```

5.3.3 The DDQN Architecture

In the said MDP problem, the state space is continuous because the channel gain $g^t_{i,u}$ to users and the queue length q^t_i of AP i have continuous values. This means it is impossible to update the Q-value for each state and action individually. Therefore, it is necessary to have an efficient way to update Q-values. In addition, the agent has to be aware of vulnerable/valuable states, in which it is likely to receive poor/high rewards in the future. This is critical because, for example, in a state where an AP queue is nearly full, taking an action to quickly reduce the queue length is necessary. Otherwise, an overflow may occur in the future. However, in most states, e.g., when an AP has low traffic loads, the choice of actions may not result in a negative reward. By contrast, conventional Q-learning only learns the maximum Q-values without seeing the key insight behind Q-values, i.e., the importance of states and actions.

To address the said issues, Q-value $Q(y_t, v_t; \theta)$ is approximated by a neural network, i.e., DDQN [25], where θ is the weight of the neural network. Specifically, after the neural network is fed with the state, it outputs corresponding Q-values. In

DDQN, Q-values are represented by two independent parts, value of the state $V(y_t)$ and value of the action $A(y_t, v_t)$. The state value $V(y_t)$ is a scalar defined as

$$V^{\pi}(y_t) = \mathbb{E}_{v_t \sim \pi(y_t)}[Q^{\pi}(y_t, v_t)]. \tag{5.36}$$

The value of actions $A(y_t, v_t)$, also known as the advantage value function [25], is a vector with dimension $|\mathcal{A}|$, which is given by

$$A^{\pi}(y_t, v_t) = Q^{\pi}(y_t, v_t) - V^{\pi}(y_t), \tag{5.37}$$

and we have $\mathbb{E}_{v_t \sim \pi(y_t)}[A^{\pi}(y_t, v_t)] = 0$. In general, the state value and the advantage value measure the goodness of a state and an action, respectively.

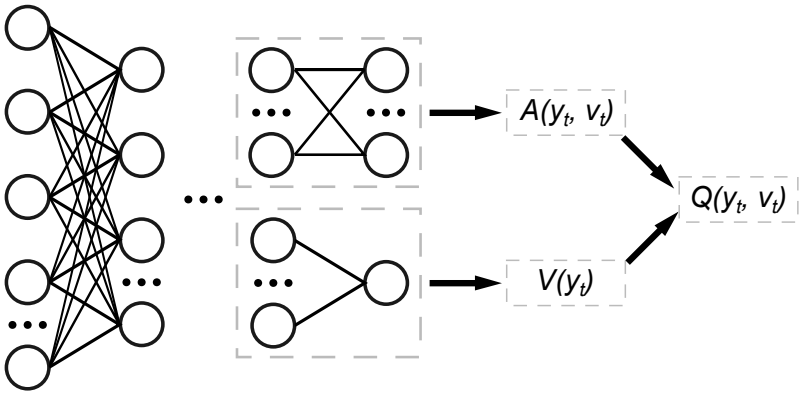

Figure 5.7: The DDQN architecture.

Define a data stream as a set of data flowing from input to output neurons. As illustrated in Fig 5.7, there are two data streams in the neural network; they estimate the state value $V(y_t; l, \theta)$ and the advantage value $A(y_t, v_t; l', \theta)$, respectively. Here, l and l' are the neuron weight of two streams, and θ represents the weight of the neural network. In the last/output layer, DDQN aggregates the state value and advantage values and then outputs Q-values. However, retrieving Q-values through the inverse operation of Eq. (5.37) result in an issue called *unidentifiability* [25]. This is because DDQN cannot determine the *unique* $V(.)$ and $A(.)$ for a given Q-value. To address this unidentifiability issue, the final Q-value will subtract a scalar $\overline{A}$ as

per [25]. Formally,

$$Q(y_t, v_t, l, l', \theta) = V^{\pi}(y_t; l, \theta) + A(y_t, v_t; l', \theta) - \overline{A}(.), \tag{5.38}$$

with

$$\overline{A}(y_t; l', \theta) = \frac{1}{|\mathcal{A}|} \sum_{v' \in \mathcal{A}} A(y_t, v'; l', \theta). \tag{5.39}$$

5.3.4 Training

This section will introduce a *target network* and *memory replay* strategy in order to train the DDQN. Instead of updating Q-values step by step as per Eq. 5.33, DDQN minimizes the average temporal difference-error of Q-values. That is,

$$L(\theta) = \min \mathbb{E}[(\hat{Q}(r_t, y_{t+1}, v_{t+1}; \theta') - Q(y_t, v_t; \theta))^2], \tag{5.40}$$

where θ' represents the neuron weight of the target network used to output targeted Q-values. The target network θ' has the same structure but different neuron weights to the evaluation network θ. For every K slots, the neuron weight of the target network θ' is replaced by that of the evaluation network θ. This ensures that the neuron weight of the evaluation network will not be changed significantly in one iteration, which improves stability when training the DDQN.

In terms of the memory replay strategy, the agent maintains a memory dataset that stores the historical state-action-reward pairs in each slot. For every K' slots, where $K' \ll K$, the agent randomly extracts a mini-batch of pairs from the dataset to train the DDQN. The Stochastic Gradient Descent method [22] with a learning rate α to minimize the loss function $L(\theta)$.

5.4 Evaluation

Simulations are conducted using Python 3.7 with TensorFlow 2.1 and MATLAB 2017. Table 5.5 and 5.3 list the parameter values used in simulations. Assume there are two APs, denoted as AP i and AP j and two users. Both users are 10 meters away from their associated AP and 18 meters away from the neighboring AP. These distances correspond to a range in which APs will use a single or bonded channel. Otherwise, an AP may use a single or bonded channel exclusively. This issue will be discussed further in experiments. Assume there are two channels C_1 and C_2, and they are adjacent. As for the DDQN agent installed at AP i, it uses a neural network that consists of four fully-connected layers and 900 neurons in total to accurately approximate Q-values. Lastly, Different learning rates α from 10^{-5} to 10^{-1} and decay rates γ from 0.1 to 0.9 are tested. The result shows that the highest reward is obtained by an agent when $\alpha = 10^{-3}$ and $\gamma = 0.7$.

The DDQN solution will be compared against the following algorithms/solutions/rules:

- **Bonded Channel Only (BCO)**: AP i will always bond channel C_1 and C_2.
- **Primary Channel Only (PCO)**: AP i will only use its primary channel C_1.
- **Random (R)**: AP i uniformly chooses an action from the action space in each slot.

Assume each episode contains 1,000 slots. The average value of the following metrics within each episode is recorded:

- **Energy Efficiency (EE)** per slot, which is calculated as $\frac{1}{1000}\sum_{t+1000}^{t}\eta_i^t$.
- **Reward**. This is the average reward earned by the agent per slot in an episode. It is calculated as $\frac{1}{1000}\sum_{t+1000}^{t} r_t$.
- **Queue length**, defined as the average number of bits queued at AP i per slot in an episode. This is calculated as $\frac{1}{1000}\sum_{t+1000}^{t} q_i^t$.

Table 5.5: Parameter values used in simulations.

Model Parameters	***Value***
Distance from a user to its associated AP	10 meters
Distance from a user to its interfering AP	18 meters
The maximum AP transmit power P_{max}	1 Watts [90]
Variance of channel gain σ^2	2 [93]
Data arrival rate λ	50 Mbps
Data queue size L	300 Mb
Total simulated time slots T	120,000

Algorithm Parameters	***Value***
Learning rate α	10^{-3}
Reward decay rate γ	0.7
Coefficient of EE β_1	0.1
Coefficient of delay β_2	0.004
Number of actions N_A	8
Size of the memory dataset	50,000
Mini-batch size	100
Time interval to update θ K'	Every four slots
Time interval to update θ' K	Every 400 slots
Neural network configuration	Four fully connected layers
Number of neurons	900
Activation function of neurons	ReLU
Initial exploration rate ϵ_0	1
Final exploration rate ϵ_T	0.02
Training start slot	6,000-th
Assessment start slot	120,000-th

- **Number of overflows**. This metric counts the number of time slots out of T where an AP's queue is full.

5.4.1 Convergence of DDQN

Fig. 5.9 and 5.8 show that both the reward and EE of DDQN increase over time. For example, the reward increases from around 0.4 at the beginning to 1.54 after 60 episodes. This is because DDQN is able to learn a policy that returns the optimal channel and transmit power in each slot over time. By contrast, the reward and EE achieved by BCO, PCO and Random remain the same. The reward gained by BCO,

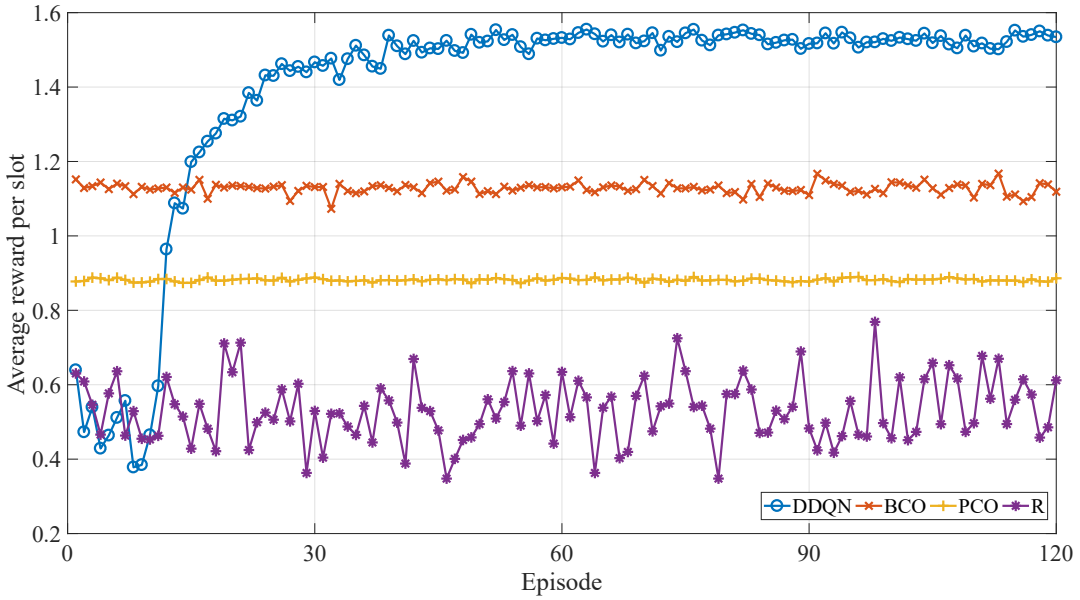

Figure 5.8: Elapsed time versus the reward gained by tested algorithms/schemes.

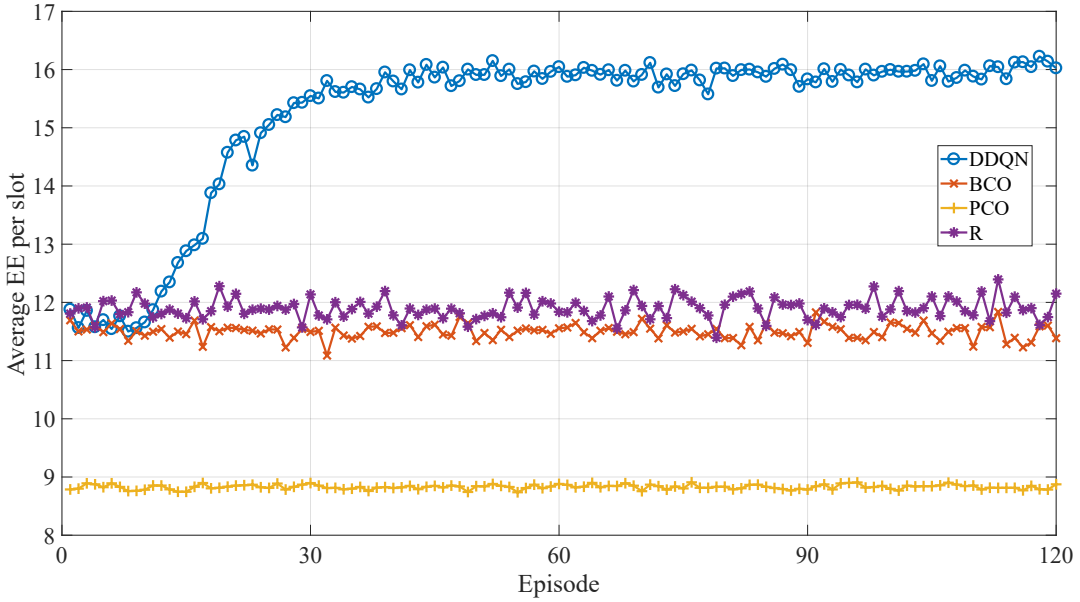

Figure 5.9: Elapsed time versus EE achieved by the tested algorithms/schemes.

PCO and Random is only 1.15, 0.88 and 0.53 over 120 episodes, respectively. As shown in Fig. 5.9, before being trained, DDQN gains only 0.4 to 0.6 reward per slot, which is similar to Random. The reason is that before the sixth episode, the agent is forced to explore the environment by taking arbitrary actions. This is to speed up convergence and avoid local optima. Then, the neural network starts updating after the sixth episode. Thus, we can see that the reward earned by DDQN increases and then peaks at 1.54 at the 60-th episode. In addition, we also see that the reward does not change after 60 episodes. This indicates that DDQN has converged after 60 episodes of training. From Fig. 5.8, we see that the EE achieved by DDQN also increases from 12 to 16 at the 60-th episode. This shows that the DDQN agent has learned to increase EE. By contrast, the EE achieved by Random, BCO and PCO remains at 11.88, 11.49 and 8.83, respectively.

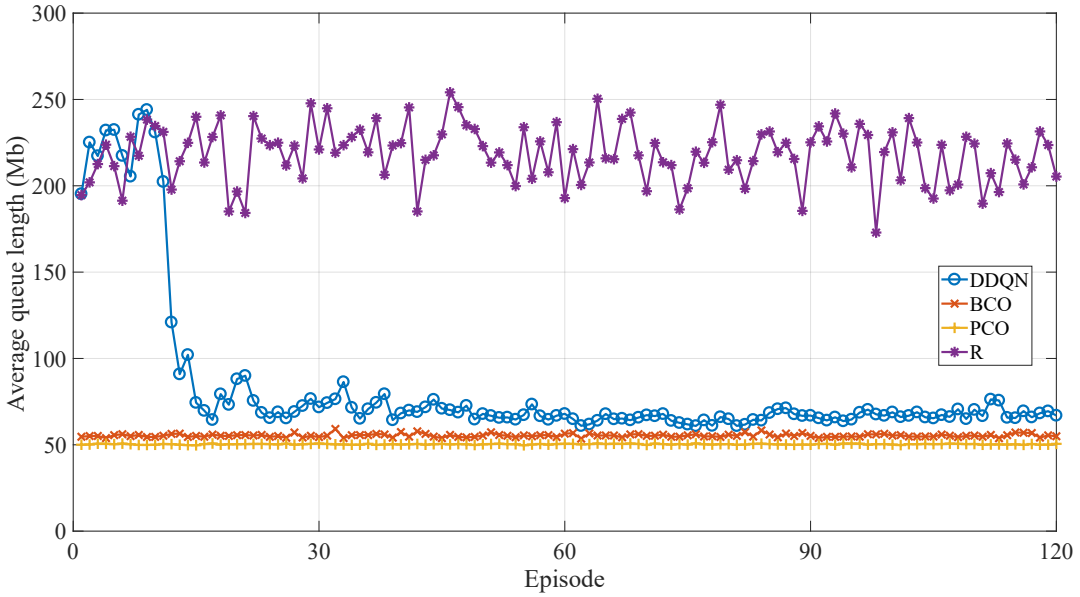

Figure 5.10: Elapsed time versus queue length achieved by the tested algorithms/schemes.

Fig. 5.10 shows that the queue length (in Mb) achieved by DDQN reduces from around 225 to 65 Mb after convergence. The queue length achieved by PCO and BCO is around 50 and 55, respectively. More importantly, from the results shown in Fig. 5.8 and 5.10, the EE achieved by DDQN is approximately 40% and 78% higher than BCO and PCO. This increase in EE is at the expense of only 20% and 25%

longer queue as compared to PCO and BCO. These results confirm that DDQN is able to balance the trade-off between EE and queuing delays, with a queue length of only 65 Mb and EE of 16 after training. In contrast, the queue length when using Random is around 225 MB; this results in a 350% longer queue than PCO and BCO. This means AP i experiences much higher queuing delay.

We also record the number of overflows. For BCO, the number of overflows at AP j is 13,273. This is caused by the strong interference experienced by AP j. By contrast, the total number of overflows at both APs is zero when using PCO and DDQN. Lastly, the number of overflows achieved by Random is 3,166. This is because AP i cannot learn to reduce interference. Hence, AP i does not have sufficient capacity to transmit data, resulting in queue overflows.

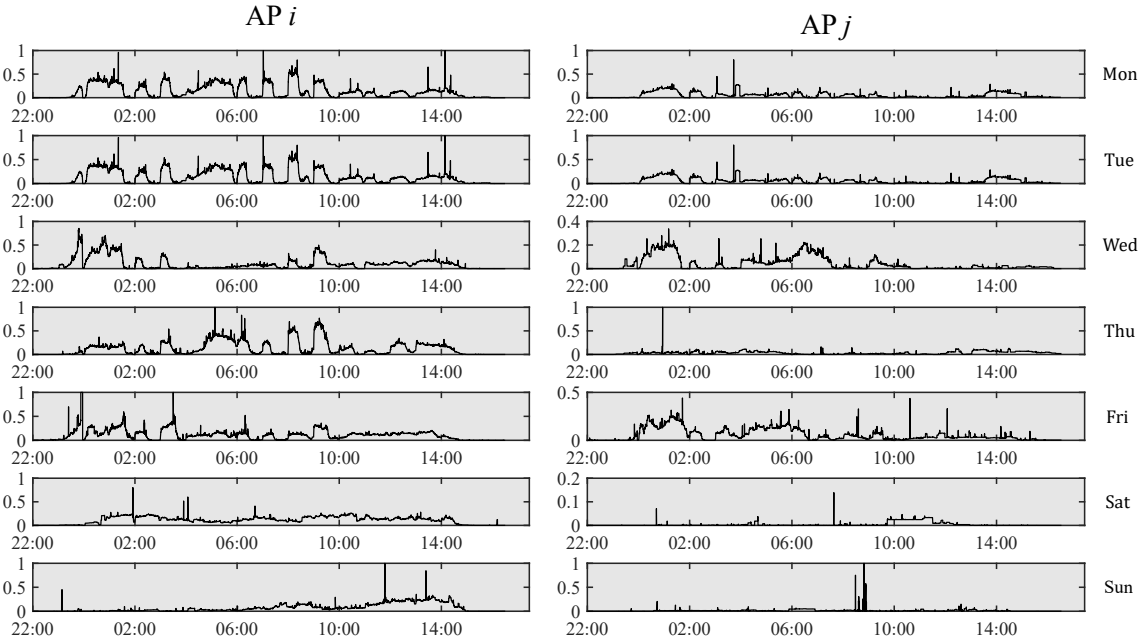

Figure 5.11: Traffic load of two APs over seven days. The Y-axis represents the normalized traffic load in each slot. The traces from [1] on Monday, Wednesday, Friday and Sunday is used to train the agent, and those on Tuesday, Thursday and Saturday is used to assess the agent.

5.4.2 Realistic Traffic

Next, DDQN will be validated using trace-based simulation. Specifically, this simulation uses traffic traces from [1], which contain the start time, end time, total

number of bytes of each HTTP session from September 2014 to January 2015. Before using these traffic traces, they need to convert the session-level information into the traffic load per unit of time. First, the traffic within a session is assumed to be constant. That is, the number of arrived bytes per second in a session does not change, which is calculated as the ratio of the total number of transmitted bytes and the duration of a session. However, an AP may serve multiple HTTP sessions at the same time. This means within one second of time, the bits arriving at an AP may come from multiple HTTP sessions at the same time. Thus, the total number of bytes that arrive at an AP in each second can be represented by the aggregated number of bytes from these sessions, which is random and varies second by second. Also, the hours in which there is zero traffic are removed. The maximum traffic is set to 300 Mb/s because the maximum queue length is 300 Mb. The traffic arrivals of two APs from October 19, 2014 at 22:00 to October 27, 2014 at 16:30 is extracted and is now depicted in Fig. 5.11.

This experiment employs the following methodology. First, to help the DDQN agent adapt to realistic traffic loads, its memory size is enlarged to 3×10^6. Second, define $\hat{t}$ as the number of seconds elapsed since 00:00 of a day. For instance, at 2:00, $\hat{t}$ is 7200. An AP will observe $\hat{t}$ in each second, meaning that the state is now revised to the tuple $y_t = (q_i^t, g_{i,u}^t, \overline{a}_i^t, y_i^t, \hat{t})$. This helps the DDQN agent adapt to realistic traffic loads, which show strong periodicity on a daily basis; see Fig. 5.11. We can also see that the amount of traffic is strongly correlated with the time of a day due to user activities; for example, the peak hour is around 1:00 or 6:00. Third, to avoid biases, the DDQN agent will be assessed after it is trained sufficiently. To be specific, the agent is first trained using the traffic traces on Monday, Wednesday, Friday and Sunday. To verify the DDQN solution, an assessment phase is defined as a time period (day) in which the reward, EE and number of overflows achieved by the DDQN solution are recorded. Specifically, the agent is then tested using the traffic traces on Tuesday, Thursday and Saturday.

Fig. 5.12 and 5.13 show the reward and EE achieved by different solutions in

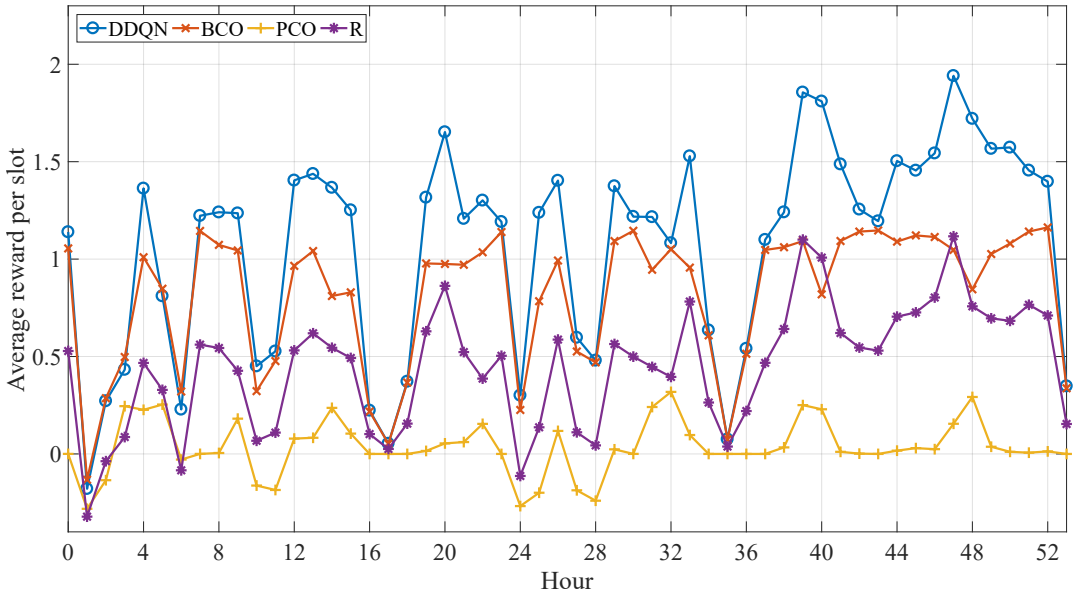

Figure 5.12: Elapsed time versus reward gained by the tested algorithms/schemes during the assessment phases.

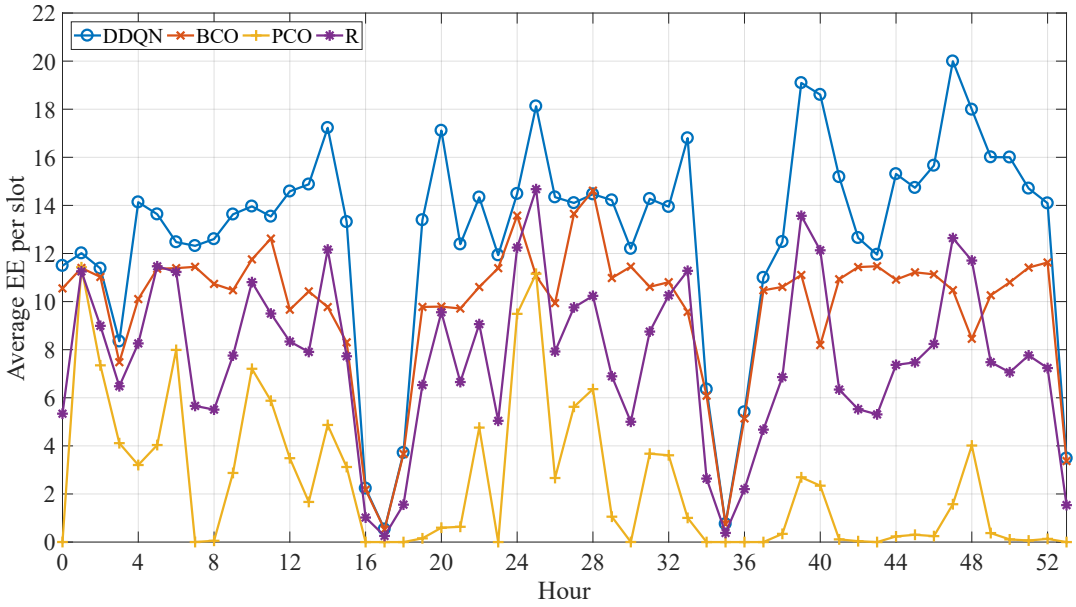

Figure 5.13: Elapsed time versus EE experienced by the tested algorithms/schemes during the assessment phases.

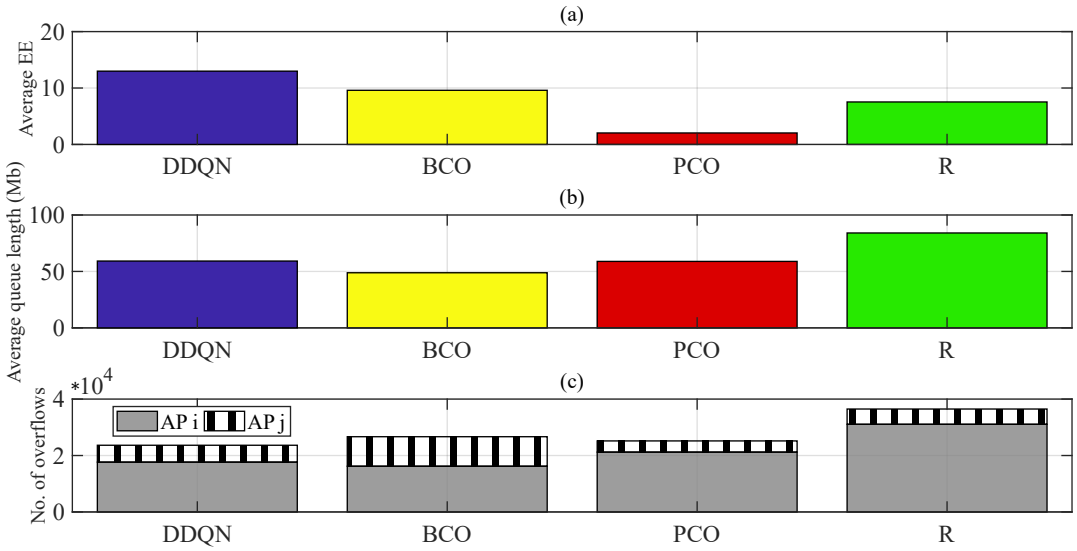

Figure 5.14: Comparisons of the average EE (a), queue length (b) and number of overflows (c) experienced by the tested algorithms/schemes during the assessment phases.

each hour during the assessment phases. The reward and the EE experienced by DDQN fluctuate from around -0.2 to 2.0 and from 0.2 to 20, respectively. This is because traffic arrivals change dramatically across peak and off-peak hours. For example, as shown in Fig. 5.11, the traffic of AP i is almost zero from around 15:00 to 23:00. By contrast, the traffic is around 150 Mb/s from 00:00 to 2:00. Further, we notice that the reward and EE at peak hours (e.g., the 40-th hour) are higher than that of off-peak hours (e.g., the 17-th hour). As discussed in Section 5.2, the reason is that an AP that transmits more data within a slot experiences higher EE. Also, at off-peak hours, an AP may have zero traffic arrival, and hence its EE is zero.

We also notice that DDQN always gains the highest reward and EE among competing solutions. For instance, from the 40-th hour to the 52-th hour, DDQN gains a reward of 1.2 to 2, whereas BCO, Random and PCO gain a reward of around 0.8, 0.4, 0.1, respectively. Referring to Fig. 5.13, DDQN always outperforms other solutions. The EE experienced by BCO ranges eight to 11, which is 45% lower than that of DDQN.

Fig. 5.14(a), 5.14(b) and 5.14(c) respectively compare the average EE, queue

length and number of overflows experienced by competing solutions during the assessment phases. Fig. 5.14(a) shows that DDQN achieves the highest average EE, with around 13. As a comparison, BCO, Random and PCO experience an average EE of 9.6, 7.5 and 2. For Random and PCO, the reason why they experience lower EE than DDQN is that they have low channel utilization. Also, an AP using PCO always uses the maximum transmit power, resulting in lower EE. By contrast, DDQN learns to adjust the transmit power of an AP according to traffic loads.

Fig. 5.14(b) shows that BCO experiences the smallest average queue length, with around 50 Mb, respectively. DDQN also reduces the queue length of an AP, with an average queue length of only 60 Mb. The queue of DDQN is only 20% longer than that of BCO. Advantageously, according to Fig. 5.14(a), DDQN achieved around 38% and 560% higher EE than BCO and PCO, respectively. This is because, after training, DDQN finds a policy that is able to balance the queue length and EE experienced by an AP.

Fig. 5.14(c) shows DDQN experiences the least number of overflows among the tested solutions. Specifically, it only experiences 23,126 overflows at both APs, whereas AP j only experiences 6,305 overflows. This is because DDQN learns to avoid interference to AP j when bonding channels. By contrast, the total number of overflows experienced by PCO, BCO and Random is respectively 8.9%, 15.3% and 57.7% more than that experienced by DDQN. For instance, BCO has a high number of overflows at both APs; i.e., AP i and j have 16,205 and 10,457 overflows, respectively. This is due to the strong interference experienced by both APs. In terms of PCO, it recorded 3,964 overflows at AP j, since there is no interference at AP j. However, the number of overflows at AP i is 21,209. Although PCO avoids interference, AP i has a low channel utilization, resulting in insufficient capacity and more overflows at peak hours. In terms of Random, AP i has an average queue length of 85 Mb and experiences 31,065 overflows. This indicates that AP i suffers the highest queuing delay when using Random. This is because Random cannot determine when to bond channels, leading to interference or low channel utilization.

5.4.3 Poisson Traffic

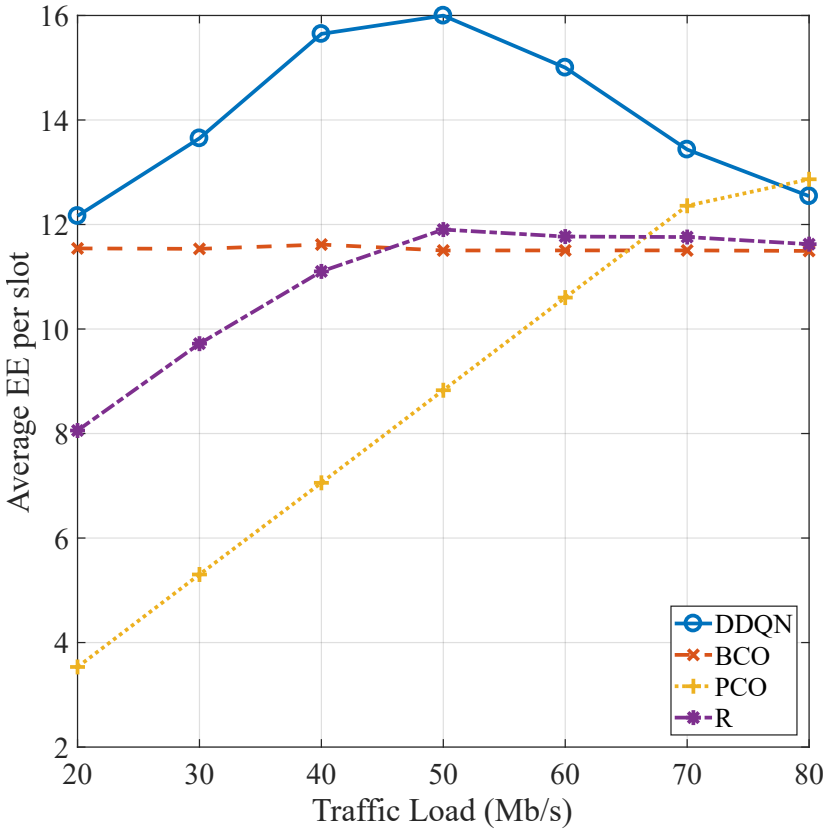

Figure 5.15: Average EE.

This experiment considers Poisson traffic. The traffic arrival rate of APs will vary between 20 and 80 Mb/s. Fig. 5.15 shows that DDQN obtains the highest EE. In particular, when the load is 50 Mb/s, the EE of DDQN is 16, which outperforms Random, BCO and PCO by 35%, 40% and 78%, respectively. As the traffic intensity of APs increases from 20 to 80 Mb/s, we can see from Fig. 5.15 that the EE gained by DDQN first increases from 12 to 16 and then decreases back to 12. The EE experienced by Random and PCO also rises from 8 to 12 and from 3.5 to 13, respectively. This is because when the traffic intensity of AP i increases to 50 Mb/s, AP i can transmit more bits in each slot. As discussed in Section 5.2, transmitting more bits within one slot yields higher EE because an AP has a fixed circuit consumption cost. In addition, AP i becomes more unlikely to be idle and experience zero energy efficiency in a slot, resulting in higher average EE. However, as the traffic intensity of APs exceeds 50 Mb/s, we see that the energy efficiency

gained by DDQN decreases from 16 to 12. One reason is that both AP i and j need to occupy the channel for a longer time. AP i receives increasing interference from AP j, which reduces the capacity of AP i. Another reason is that AP i needs to increase the transmit power and consume more energy to overcome interference. These two reasons lead to a decrease in EE of AP i. Also, Fig. 5.15 shows that the EE gained by PCO linearly grows from 3.5 to 12.5 before the traffic intensity of APs reaches 70 Mb/s.

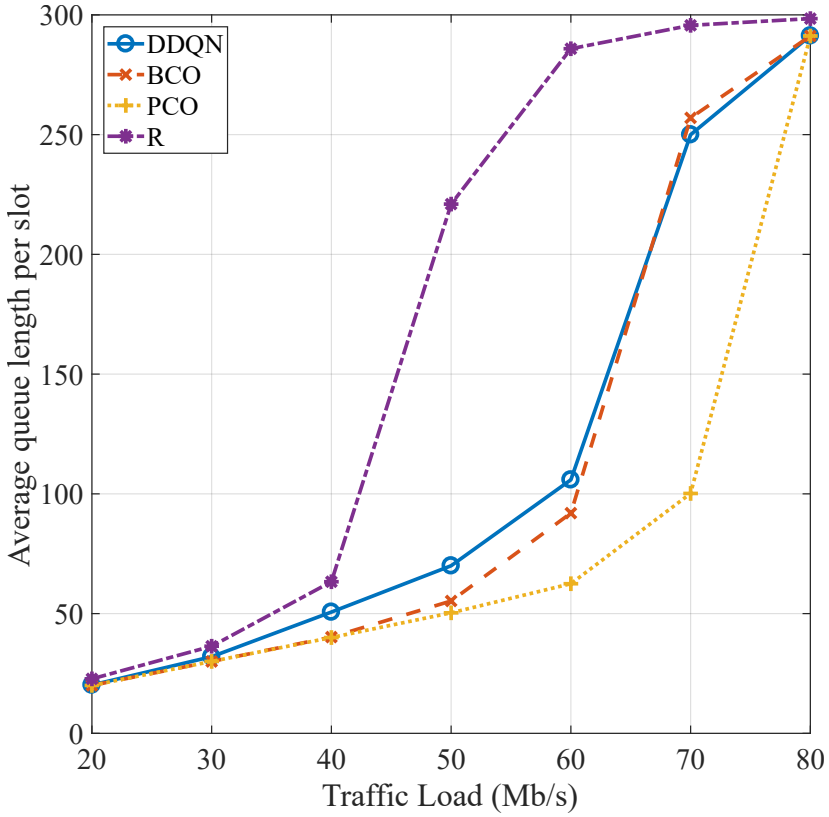

Figure 5.16: Average queue length (in Mb).

Referring to Fig. 5.15 5.16 and 5.17, at 80 Mb/s, both PCO and DDQN experience the same EE, queue length and number of overflows. For instance, the EE experienced by both PCO and DDQN is 12. This means DDQN learns to use only a primary channel when the traffic intensity of APs is 80 Mb/s. From Fig. 5.16 and 5.17, all competing solutions have longer queues and higher overflows when traffic increases from 60 to 80 Mb/s. For example, the queue length experienced by DDQN increases from 100 to near 300 Mb. Also, the number of overflows experi-

enced by DDQN increases from 0 to 30,000.

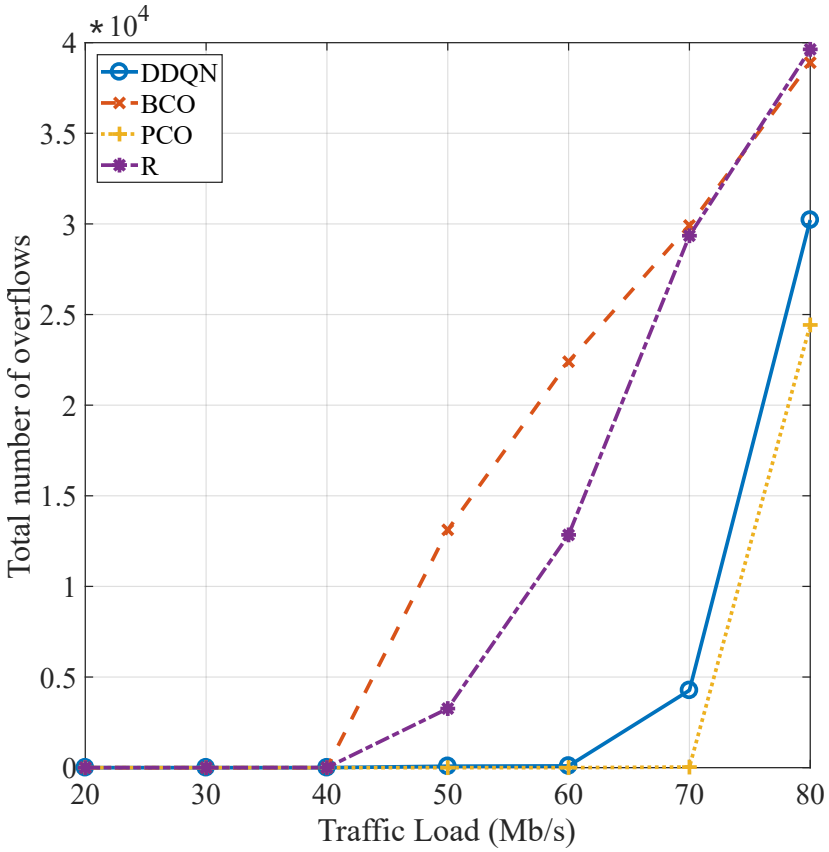

Figure 5.17: Total number of overflows.

The DDQN agent receives negative rewards because of longer queue length and more overflows. Consequently, DDQN learns to use a higher transmit power, which causes the EE reduction shown in Fig. 5.15 when the traffic intensity exceeds 50 Mb/s. Further, in Fig. 5.17, once the traffic intensity exceeds 70 Mb/s, the number of overflows experienced by APs sharply rises to 30,000 when using DDQN and PCO. This is because AP i does not have sufficient capacity when the traffic intensity is 70 Mb/s. Therefore, balancing the trade-off between EE and queuing delay experienced by AP i is not practical when the traffic intensity of APs is greater than 70 Mb/s.

5.4.4 Neighbor Distances

In this experiment, the distance between a user and its *associated* AP is always 10 meters; e.g., the distance from user u to AP j. However, the distance between a user

and its *neighboring/interfering* AP will vary from 10 to 22 meters; i.e., the distance from user u to AP j and from user v to AP i. The reason for choosing 10 to 22 meters is that we can clearly see how the different distances to an interfering AP change the channel bonding policy used by AP i.

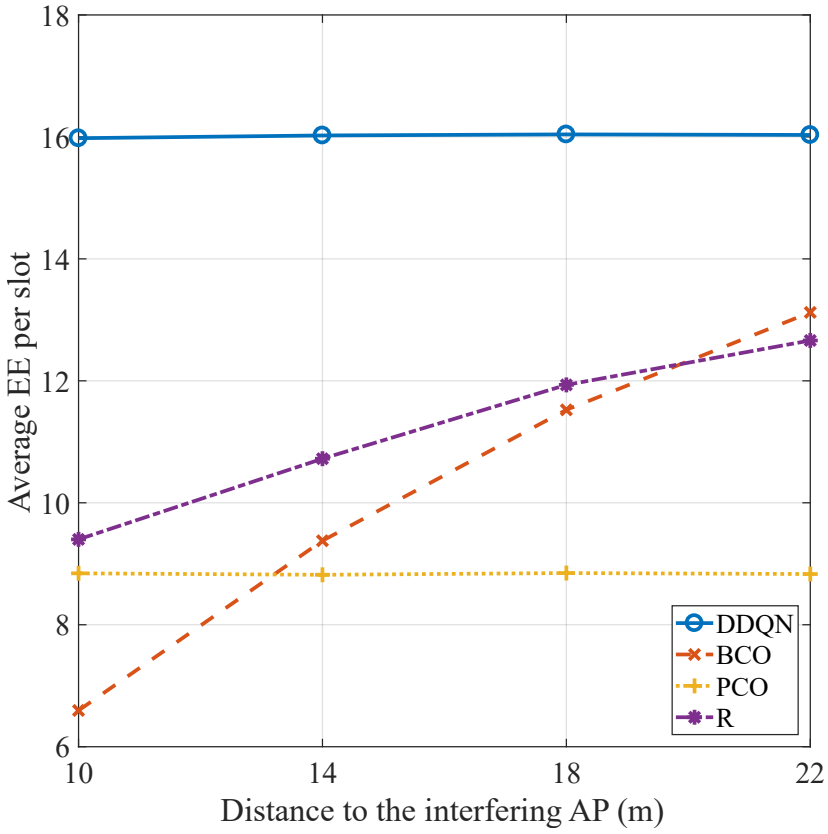

Figure 5.18: Average EE.

Referring to Fig. 5.18, we see that DDQN always gains the highest EE. Specifically, DDQN achieves at least 23%, 28% and 83% higher EE than BCO, PCO and Random achieve, respectively. From Fig. 5.18, 5.19 and 5.20, we see that the EE, queue length and number of overflows experienced by DDQN are not affected by the decrease in the distance to an interfering AP. For example, the EE gained by DDQN is always around 16. This is because DDQN always learns the optimal channel and transmit power policy when a user is at different distances to its interfering AP. In contrast, the EE experienced by BCO and Random respectively declines from 12.5 to 9.5 and from 13 to 6.5 as the distance to an interfering AP decreases from 22 to 10 meters. This is because users experience a higher level of interference. This

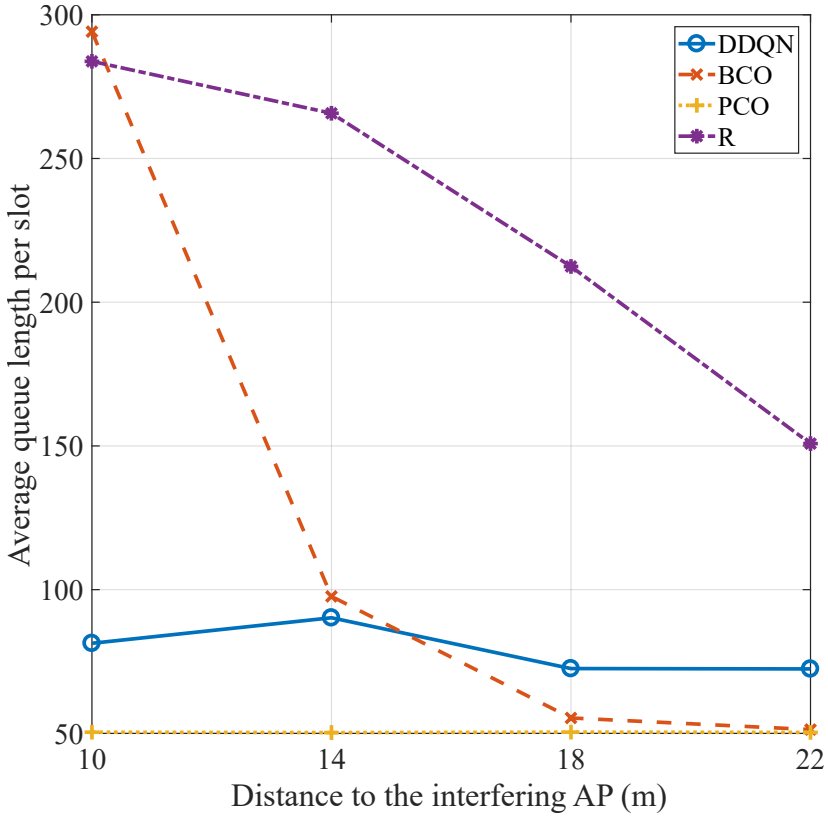

Figure 5.19: Average queue length (in Mb).

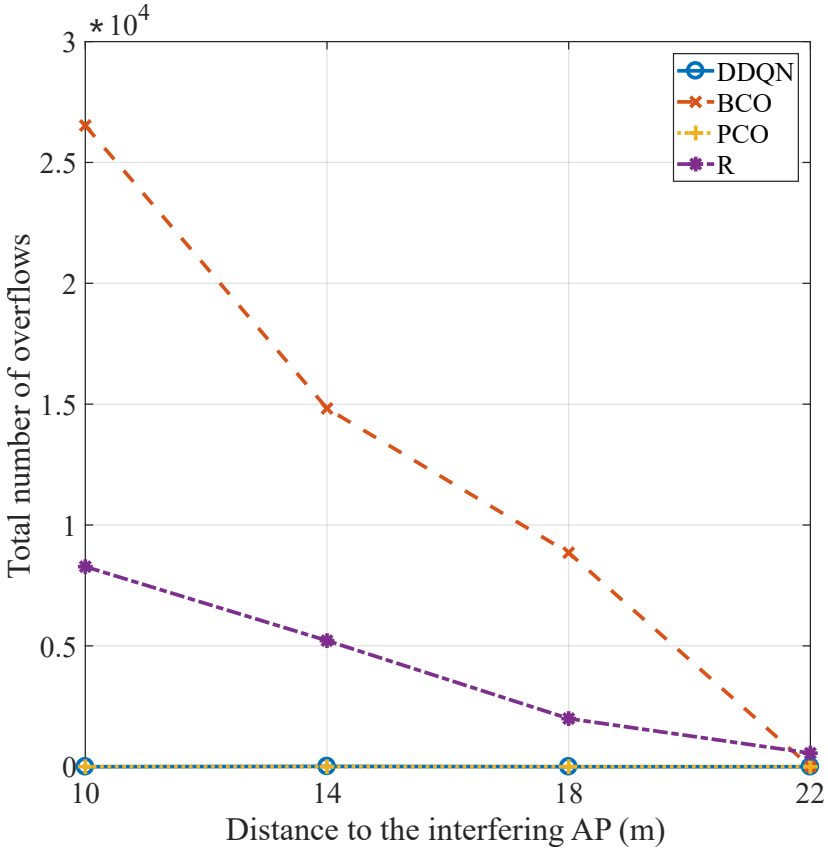

Figure 5.20: Total number of overflows.

also shows that using a bonded channel becomes harmful to EE as the distance to an interfering AP decreases. In addition, we see from Fig. 5.18, 5.19 and 5.20 that the EE, queue length and number of overflows experienced by PCO remains stable as the distance to an interfering AP decreases. For instance, the EE achieved by PCO is around 8.8. This is because two APs are assigned different channels when using PCO and thus users do not experience interference at any distance. However, the result in Fig. 5.18 shows that when the distance to an interfering AP is greater than 14 meters, PCO is no better than BCO because PCO cannot utilize channel resources effectively.

Fig. 5.19 shows that DDQN reduces the queue length experienced by AP i to under 90 Mb. It is shorter than the queue for Random by 40% to 67%. When AP i uses BCO, it has at most 25% shorter queue length than when it uses DDQN after the distance to an interfering AP is greater than 14 meters. The queue length experienced by BCO is almost four times that of DDQN, meaning using DDQN experiences much lower queuing delay than using BCO for distances smaller than 14 meters. PCO experiences the smallest queuing delay, where the queue length is 50 Mb. However, the EE of PCO is 45% less than DDQN. This is because an AP with PCO does not bond channels to obtain a high data rate. Also, PCO always uses the maximum transmit power, leading to lower EE.

Referring to Fig. 5.20, APs do not experience queue overflow when they use DDQN and PCO. This is because DDQN learns to minimize the interference at both APs to ensure zero overflows at AP j. In contrast, Random and BCO lead to more overflows as the distance to an interfering AP decreases. The number of overflows for Random and BCO rises to 7,500 and 26,000. This is because AP i produces stronger interference to AP j. Hence, AP j has insufficient capacity. Moreover, Fig. 5.20 shows that the number of overflows for BCO is almost four times as compared to that for Random when the distance to an interfering AP is less than 10 meters. At 10 meters, using BCO leads to poor performance due to strong interference between APs.

5.5 Conclusion

This chapter aims to improve the EE of an AP and also reduce its queue length. According to the mathematical analysis, the EE experienced by an AP is closely related to the channel bonding policy and transmit power level adopted by the AP. This chapter has proposed a DRL solution to determine the channels and transmit power level for an AP. Advantageously, it does not require information about interference and traffic loads. Numerical results show that the proposed DRL solution is able to learn the optimal channel bonding and transmit power allocation policy that improves the EE experienced by an AP by up to 560% as compared to using a fixed or random policy.

Chapter 6

Conclusion

This thesis has conducted a comprehensive study on introducing cognition in WLANs. Specifically, it proposes numerous machine-learning approaches for WLANs. The key aim is to satisfy the data rate of users or energy requirement of sensor devices. As shown in this thesis, these approaches assign channels and a transmit power level to APs yielding the minimum interference and the maximum EE. However, a key problem is that user demands, the energy arrivals to an AP and CSI are random and vary over time, meaning that any solution must be adaptive. Critically, APs may not have perfect or non-causal information about the said quantities. Another problem is that APs may experience increasing interference when switching to a bonded channel or using a high transmit power level. However, existing works that consider this problem assume APs known perfect CSI or traffic loads beforehand. In addition, they only optimize for a given amount of traffic load.

To this end, this thesis contains a number of resource allocation solutions for Wi-Fi networks. Unlike existing works, the proposed machine-learning approaches have a learning ability and do not rely on perfect and non-causal information, e.g., perfect CSI and non-causal traffic arrivals. In a nutshell, it makes the following contributions:

- To deal with spatio-temporal traffic [15], Chapter 3 addresses the channel

bonding problem where APs have random traffic demands. It first formulates this problem as an MDP that defines the state of an AP, the channel(s) used by the AP, and resulting rewards for using one or more channel(s). Then, it presents a DRL solution that allows each AP to independently learn to bond one or more channel(s) in order to satisfy its traffic demands while minimizing the interference to neighboring APs. Unlike existing works, the solution only relies on historical traffic demands, whereas current traffic demands are unknown to an AP. This makes the proposed solutions practical and readily deployable by current APs. Simulation results show that the proposed DRL solution has up to 60% more user satisfaction than greedy and fixed channel bonding algorithms.

- Future Wi-Fi systems may comprise of RF-energy harvesting devices, e.g., temperature sensors and on-board cameras [10]. These devices may be equipped with an RF-energy harvestor that can be charged whenever an AP transmits to its data users. This means operators do not need to deploy dedicated energy beacons for these devices. In this regard, Chapter 5 considers a novel problem that an AP has to support RF-energy harvesting devices while satisfying the data rate requirement of legacy data users. One issue is that an AP has only *causal* energy arrivals, meaning that it has to optimize the use of energy. Another issue is that an AP has no information of *current* CSI to devices. This is because obtaining the CSI will affect data users and consume extra energy. Existing works that consider two types of users have not addressed the said issues. To this end, Chapter 5 proposes two machine-learning solutions, i.e., DQN and MPC, to determine the optimal transmit power policy for an AP. This policy aims to satisfy both the energy and data rate requirement of users at the same time. Both solutions do not require perfect CSI to devices and they are adaptable to time-varying energy arrival rates. In addition, both solutions are readily deployable in current APs. Numerical results show that

the proposed DQN and MPC algorithms are able to support up to 42% more IoT devices and achieve up to 27% more user satisfaction than competing algorithms.

- To address the concern of EE [8], Chapter 4 considers EE optimization under random traffic and interference. Specifically, an AP has to determine its channel and transmit power in order to maximize its EE and avoid interference with its neighboring APs. A key challenge is that an AP has no information of the traffic load of its neighboring APs, meaning that it is not aware of the amount of interference when using a bonded channel. For example, it does not know when does its neighbor has no data to transmit. Another key problem is that the traffic load is realistic and causal. Past works on EE optimization for Wi-Fi systems only consider either transmit power control or channel bonding. Critically, they do not consider traffic variations or random channel gain. To this end, this chapter proposes a solution based on DDQN [25] to learn the optimal channel bonding and transmit power policy for an AP. The learned policy takes queue delay, overflow and EE experienced by the AP into consideration. The said problem is first studied using a novel six-state CTMC, which shows some insight into the use of channel bonding and transmit power control. Then, the problem is formulated as a model-free MDP and is solved by DDQN. Numerical results show that the proposed DDQN solution is able to improve the EE experienced by an AP by up to 560% at the least expense of long queues or overflows as compared to using a fixed or random policy.

There are a number of possible future works. First, for the problem of charging RF-energy harvesting devices in Wi-Fi networks, Chapter 5 only considers one energy source, i.e., the AP. However, this problem can be extended by considering using multiple APs to charge devices. These APs are able to switch to different channels and transmit power levels. The problem is to minimize the interference among APs while guaranteeing that devices harvest sufficient energy. Indeed, RF-

energy harvesting devices benefit from interference while legacy data users not. For the same reason, the problem in Chapter 4 can also be extended by considering EE optimization for multiple APs, where all APs have the ability to vary their channel and transmit power level. Also, the work in Chapter 3 can be extended by considering AP switching on/off, where APs can be switched off when they have no user. This further improves the energy efficiency of the APs. Another future direction is to incorporate data on users behaviors to optimize a wireless network. This data includes user distribution, spatio-temporal traffic or type of service. Such data can be collected by internet service providers. Given this data, a machine learning algorithm can then be used to jointly optimize network parameters such as transmit power, channel, modulation scheme or antenna direction. Third, a future research direction is to consider transfer learning to allow an agent that is trained in one environment to be used in a different environment.

Bibliography

[1] J. Liu, B. Krishnamachari, S. Zhou, and Z. Niu, "DeepNap: Data-driven base station sleeping operations through deep reinforcement learning," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4273–4282, 2018.

[2] Y.-C. Chen, J. Kurose, and D. Towsley, "A mixed queueing network model of mobility in a campus wireless network," in *IEEE INFOCOM*, (Orlando, FL, USA), pp. 2656–2660, Mar. 2012.

[3] O. Lee, J. Kim, and S. Choi, "WiZizz: Energy efficient bandwidth management in IEEE 802.11ac wireless networks," in *IEEE SECON*, (Seattle, USA), pp. 136–144, June 2015.

[4] A. Gupta and R. K. Jha, "A survey of 5G network: Architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, July 2015.

[5] "Improving energy efficiency, lower CO_2 emission and TCO." White Paper, 2009.

[6] D. Feng, C. Jiang, G. Lim, L. J. Cimini, G. Feng, and G. Y. Li, "A survey of energy-efficient wireless communications," *IEEE Commun. Surveys Tuts.*, vol. 15, pp. 167–178, Feb. 2013.

[7] L. Xu, K. Yamamoto, and S. Yoshida, "Performance comparison between

channel-bonding and multi-channel CSMA," in *IEEE Wireless Commun. and Networking Conf.*, pp. 406–410, Mar. 2007.

[8] K. Wang, Y. Wang, Y. Sun, S. Guo, and J. Wu, "Green industrial internet of things architecture: An energy-efficient perspective," *IEEE Comms. Mag.*, vol. 54, pp. 48–54, Dec. 2016.

[9] T. Wang, C. Jiang, and Y. Ren, "Access points selection in super WiFi network powered by solar energy harvesting," in *IEEE WCNC*, (Doha, Qatar), pp. 1–6, Apr. 2016.

[10] V. Talla, B. Kellogg, B. Ransford, S. Naderiparizi, S. Gollakota, and J. R. Smith, "Powering the next billion devices with Wi-Fi," in *ACM Conf. on Emerging Netw. Experiments and Technol.*, (Heidelberg, Germany), Dec. 2015.

[11] B. Bellalta, "IEEE 802.11ax: High-efficiency WLANs," *IEEE Wireless Commun.*, vol. 23, pp. 38–46, Feb. 2016.

[12] S. Khairy, M. Han, L. X. Cai, and Y. Cheng, "Sustainable wireless IoT networks with RF energy charging over Wi-Fi (CoWiFi)," *IEEE Internet of Things J.*, vol. 6, pp. 10205–10218, Dec. 2019.

[13] L. Deek, E. Garcia-Villegas, E. Belding, S. Lee, and K. Almeroth, "Intelligent channel bonding in 802.11n WLANs," *IEEE Trans. Mobile Comput.*, vol. 13, pp. 1242–1255, June 2014.

[14] M. Min, L. Xiao, Y. Chen, P. Cheng, D. Wu, and W. Zhuang, "Learning-based computation offloading for IoT devices with energy harvesting," *IEEE Trans Vehic. Tech.*, vol. 68, pp. 1930 – 1941, Feb. 2019.

[15] L. Yang, L. Cao, H. Zheng, and E. Belding, "Traffic-aware dynamic spectrum access," in *ACM WICON*, (Maui, Hawaii, USA), Nov. 2008.

[16] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957.

[17] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, USA: MIT press, 2018.

[19] S. Rayanchu, V. Shrivastava, S. Banerjee, and R. Chandra, "FLUID: Improving throughput in enterprise wireless LANs through flexible channelization," *IEEE Trans. Mobile Comput.*, vol. 11, pp. 1455–1469, Sept. 2012.

[20] W. Yuan, P. Wang, W. Liu, and W. Cheng, "Variable-width channel allocation for access points: A game-theoretic perspective," *IEEE Trans. Mobile Comput.*, vol. 12, pp. 1428–1442, July 2013.

[21] A. Mishra, V. Shrivastava, S. Banerjee, and W. Arbaugh, "Partially overlapped channels not considered harmful," *SIGMETRICS Perform. Eval. Rev.*, vol. 34, pp. 63–74, June 2006.

[22] L. Yann, B. Yoshua, and H. Geoffrey, "Deep learning," *Nature*, vol. 521, p. 436–444, 2015.

[23] M. Morari, C. E. Garcia, and D. M. Prett, "Model predictive control: Theory and practice," *IFAC Proceedings Volumes*, vol. 21, pp. 1–12, June 1988.

[24] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[25] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of Machine Learning Research*, vol. 48, (New York, USA), pp. 1995–2003, June 2016.

[26] C. Kai, Y. Liang, T. Huang, and X. Chen, "To bond or not to bond: An optimal channel allocation algorithm for flexible dynamic channel bonding in WLANs," in *IEEE Veh. Technol. Conf.*, (Toronto, ON, Canada), pp. 1–6, Sept. 2017.

[27] S. Barrachina-Muñoz, F. Wilhelmi, and B. Bellalta, "Performance analysis of dynamic channel bonding in spatially distributed high density WLANs," *CoRR*, 2018.

[28] W. Wang, F. Zhang, and Q. Zhang, "Managing channel bonding with clear channel assessment in 802.11 networks," in *IEEE Int. Conf. on Commun. (ICC)*, (Kuala Lumpur, Malaysia), pp. 1–6, May 2016.

[29] T. Song, T. Y. Kim, W. Kim, and S. Pack, "Adaptive and distributed radio resource allocation in densely deployed wireless LANs: A game-theoretic approach," *IEEE Trans. Veh. Technol.*, vol. 67, pp. 4466–4475, May 2018.

[30] T. Song and T. Y. Kim and W. Kim and S. Pack, "Channel bonding algorithm for densely deployed wireless LAN," in *Int. Conf. on Information Networking (ICOIN)*, (Kota Kinabalu, Malaysia), pp. 395–397, Jan. 2016.

[31] R. Chandra, R. Mahajan, T. Moscibroda, R. Raghavendra, and P. Bahl, "A case for adapting channel width in wireless networks," *ACM SIGCOMM*, vol. 38, pp. 135–146, Aug. 2008.

[32] A. Nabil, M. J. Abdel-Rahman, A. B. MacKenzie, and F. Hassan, "A stochastic optimization framework for channel bonding in wireless LANs under demand uncertainty," *IEEE Trans. Wireless Commun.*, vol. 19, pp. 7528–7542, Nov. 2020.

[33] T. Moscibroda, R. Chandra, Y. Wu, S. Sengupta, P. Bahl, and Y. Yuan, "Load-aware spectrum distribution in wireless LANs," in *IEEE Int. Conf. on Network Protocols*, (Orlando, FL, USA), pp. 137–146, Oct. 2008.

[34] F. Zarinni and S. R. Das, "Adaptive spectrum distribution in WLANs," in *IEEE GLOBECOM*, (Miami, FL, USA), pp. 1–6, Dec. 2010.

[35] H. Qi, H. Huang, Z. Hu, X. Wen, and Z. Lu, "On-demand channel bonding in heterogeneous WLANs: A multi-agent deep reinforcement learning approach," *Sensors*, pp. 1–16, Dec. 2020.

[36] X. Lu, P. Wang, D. Niyato, D. I. Kim, and Z. Han, "Wireless networks with RF energy harvesting: A contemporary survey," *IEEE Commun. Surveys & Tut.*, vol. 17, no. 2, pp. 757–789, 2015.

[37] K.-W. Chin, "On energy and data delivery in wireless local area networks with RF charging nodes," in *IEEE ITNAC*, pp. 1–7, Nov. 2017.

[38] M. Z. Sarwar and K. Chin, "On supporting legacy and RF energy harvesting devices in two-tier OFDMA heterogeneous networks," *IEEE Access*, vol. 6, pp. 62538–62551, 2018.

[39] M. A. Abd-Elmagid, T. ElBatt, and K. G. Seddik, "Optimization of wireless powered communication networks with heterogeneous nodes," in *IEEE GLobecom*, (San Diego, CA, USA), Dec. 2015.

[40] T. Wu and H. Yang, "RF energy harvesting with cooperative beam selection for wireless sensors," *IEEE Wireless Commun. Lett.*, vol. 3, pp. 585–588, Dec. 2014.

[41] J. Ebert, B. Stremmel, E. Wiederhold, and A. Wolisz, "An energy-efficient power control approach for WLANs," *Journal of Commun. and Netw.*, vol. 2, pp. 197–206, Sept. 2000.

[42] O. Oteri, Pengfei Xia, F. LaSita, and R. Olesen, "Advanced power control techniques for interference mitigation in dense 802.11 networks," in *WPMC*, (Atlantic City, USA), pp. 1–7, June 2013.

[43] Yang Wu, Yongmei Sun, Yuefeng Ji, Jie Mao, and Yingting Liu, "A joint channel allocation and power control scheme for interference mitigation in high-density WLANs," in *15th IEEE Int. Conf. on Communication Technology*, (Guilin, China), pp. 98–103, Nov. 2013.

[44] K. Lee and J. Hong, "Power control for energy efficient D2D communication in heterogeneous networks with eavesdropper," *IEEE Commun. Letters*, vol. 21, pp. 2536–2539, Nov. 2017.

[45] Y. Zeng, P. H. Pathak, and P. Mohapatra, "A first look at 802.11ac in action: Energy efficiency and interference characterization," in *IFIP Networking Conf.*, (Trondheim, Norway), pp. 1–9, June 2014.

[46] R. Zhu and J. Wang, "Power-efficient spatial reusable channel assignment scheme in WLAN mesh networks," *Mobile Networks and Applications*, vol. 17, pp. 53–63, Mar. 2012.

[47] Z. Han, C. Xu, G. Zhao, R. An, X. Wang, and J. Zhou, "Joint optimization of energy efficiency and interference for green WLANs," in *Int. Conf. on Commun. and Netw. in China*, (Chengdu, China), pp. 204–213, Oct. 2018.

[48] Q. Wu, W. Chen, M. Tao, J. Li, H. Tang, and J. Wu, "Resource allocation for joint transmitter and receiver energy efficiency maximization in downlink OFDMA systems," *IEEE Trans. Commun.*, vol. 63, pp. 416–430, Feb. 2015.

[49] G. Zhao, Q. Wang, C. Xu, and S. Yu, "Analyzing and modelling the interference impact on energy efficiency of WLANs," in *IEEE ICC*, (Kansas City, USA), pp. 1–6, July 2018.

[50] V. P. Mhatre, K. Papagiannaki, and F. Baccelli, "Interference mitigation through power control in high density 802.11 WLANs," in *IEEE INFOCOM*, (Barcelona, Spain), pp. 535–543, May 2007.

[51] M. F. Tuysuz, "Towards providing optimal energy-efficiency and throughput for IEEE 802.11 WLANs," *Int. Journal of Communication Systems*, vol. 31, June 2018.

[52] Y. Zhang, C. Jiang, Z. Han, S. Yu, and J. Yuan, "Interference-aware coordinated power allocation in autonomous Wi-Fi environment," *IEEE Access*, vol. 4, pp. 3489–3500, June 2016.

[53] S. A. Borbash and A. Ephremides, "Wireless link scheduling with power control and SINR constraints," *IEEE Trans. Inf. Theory*, vol. 52, pp. 5106–5111, Oct. 2006.

[54] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 2019.

[55] S. Bhattacharjee, A. Bhar, and R. Saha, "Channel allocation in a cognitive radio network using non deterministic Q learning algorithm," in *3th Int. Conf. on Emerging Applications of Information Technology*, (Kolkata, India), pp. 327–330, Nov. 2012.

[56] X. Li, J. Fang, W. Cheng, H. Duan, Z. Chen, and H. Lia, "Intelligent power control for spectrum sharing in cognitive radios: A deep reinforcement learning approach," *IEEE Access*, pp. 1–1, Apr. 2018.

[57] F. Shah-Mohammadi and A. Kwasinski, "Deep reinforcement learning approach to QoE-driven resource allocation for spectrum underlay in cognitive radio networks," in *IEEE Int. Conf. on Commun. Workshops*, (Kansas City, MO, USA), pp. 1–6, May 2018.

[58] M. Bennis and D. Niyato, "A Q-learning based approach to interference avoid-

ance in self-organized femtocell networks," in *IEEE Globecom Workshops*, (Miami, FL, USA), pp. 706–710, Dec. 2010.

[59] A. Galindo-Serrano and L. Giupponi, "Distributed Q-learning for interference control in ofdma-based femtocell networks," in *IEEE Veh. Technol. Conf.*, (Taipei, Taiwan), pp. 1–5, May 2010.

[60] H. Saad, A. Mohamed, and T. ElBatt, "A cooperative Q-learning approach for distributed resource allocation in multi-user femtocell networks," in *IEEE Wireless Commun. and Networking Conf.*, (Istanbul, Turkey), pp. 1490–1495, Apr. 2014.

[61] J. Nie and S. Haykin, "A Q-learning-based dynamic channel assignment technique for mobile communication systems," *IEEE Trans. Veh. Technol.*, vol. 48, pp. 1676–1687, Sept. 1999.

[62] E. Ghadimi, F. D. Calabrese, G. Peters, and P. Soldati, "A reinforcement learning approach to power control and rate adaptation in cellular networks," in *IEEE Int. Conf. on Commun. (ICC)*, (Paris, France), pp. 1–7, May 2017.

[63] A. Adeel, H. Larijani, and A. Ahmadinia, "Resource management and inter-cell-interference coordination in LTE uplink system using random neural network and optimization," *IEEE Access*, vol. 3, pp. 1963–1979, Oct. 2015.

[64] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen, "Multi-objective reinforcement learning for cognitive satellite communications using deep neural network ensembles," *IEEE J. Sel. Areas Commun.*, May 2018.

[65] S. Liu, X. Hu, and W. Wang, "Deep reinforcement learning based dynamic channel allocation algorithm in multibeam satellite systems," *IEEE Access*, vol. 6, pp. 15733–15742, Feb. 2018.

[66] F. Wilhelmi, B. Bellalta, C. Cano, and A. Jonsson, "Implications of decentralized Q-learning resource allocation in wireless networks," in *IEEE 28th Annual Int. Symposium on Personal, Indoor, and Mobile Radio Commun. (PIMRC)*, (Montreal, Canada), Aug. 2017.

[67] D. Vengerov, N. Bambos, and H. R. Berenji, "A fuzzy reinforcement learning approach to power control in wireless transmitters," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 35, pp. 768–778, Aug. 2005.

[68] S. N. Yasar and G. Dongning, "Deep reinforcement learning for distributed dynamic power allocation in wireless networks," *arXiv*, Aug. 2018.

[69] R. Karmakar, S. Chattopadhyay, and S. Chakraborty, "Smartla: Reinforcement learning-based link adaptation for high throughput wireless access networks," *Computer Commun.*, vol. 110, pp. 1 – 25, 2017.

[70] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks," in *IEEE Globecom*, (Singapore), pp. 1–7, Dec. 2017.

[71] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari, "Deep reinforcement learning for dynamic multichannel access in wireless networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 4, pp. 257–265, June 2018.

[72] S. H. R. Bukhari, M. H. Rehmani, and S. Siraj, "A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 924–948, 2016.

[73] A. Nabil, M. J. Abdel-Rahman, and A. B. MacKenzie, "Adaptive channel bonding in wireless LANs under demand uncertainty," in *IEEE PIMRC*, (Montreal, QC, Canada), pp. 1–7, Oct. 2017.

[74] P. Gallo, K. Kosek-Szott, S. Szott, and I. Tinnirello, "CADWAN: A control architecture for dense WiFi access networks," *IEEE Comms. Mag.*, vol. 56, pp. 194–201, Jan. 2018.

[75] S. Chieochan, E. Hossain, and J. Diamond, "Channel assignment schemes for infrastructure-based 802.11 WLANs: A survey," *IEEE Commun. Surveys Tuts.*, vol. 12, pp. 124–136, First Quarter 2010.

[76] E. Rozner, Y. Mehta, A. Akella, and L. Qiu, "Traffic-aware channel assignment in enterprise wireless LANs," in *IEEE Int. Conf. on Network Protocols*, (Beijing, China), pp. 133–143, Oct. 2007.

[77] L. Busoniu, R. Babuska, and B. D. Schutter, "A comprehensive survey of multi-agent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 38, pp. 156–172, Mar. 2008.

[78] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[79] C. Gaskett, D. Wettergreen, and A. Zelinsky, "Q-learning in continuous state and action spaces," in *Advanced Topics in Artificial Intelligence*, (Sydney, Australia), pp. 417–428, Dec. 1999.

[80] M. Ku, Y. Chen, and K. J. R. Liu, "Data-driven stochastic models and policies for energy harvesting sensor communications," *IEEE J. Sel. Areas Commun.*, vol. 33, pp. 1505–1520, Aug. 2015.

[81] F. Alneyadi, M. Alkaabi, S. Alketbi, S. Hajraf, and R. Ramzan, "2.4 GHz WLAN RF energy harvester for passive indoor sensor nodes," in *IEEE Int. Conf. on Semiconductor Electronics*, (Kuala Lumpur, Malaysia), pp. 471–474, Aug. 2014.

[82] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *IEEE Commun. Surveys & Tut.*, vol. 13, pp. 443–461, July 2011.

[83] R. Takitoge, S. Ishigaki, T. Ishige, and K. Ishibashi, "Temperature beat: Persistent and energy harvesting wireless temperature sensing scheme," in *IEEE Sensors*, (Orlando, USA), pp. 1–3, Oct. 2016.

[84] M. Y. Arslan, K. Pelechrinis, I. Broustis, S. Singh, S. V. Krishnamurthy, S. Addepalli, and K. Papagiannaki, "ACORN: An auto-configuration framework for 802.11n WLANs," *IEEE/ACM Trans. Netw.*, vol. 21, pp. 896–909, Oct. 2013.

[85] F. Wilhelmi, S. Barrachina-Muñoz, B. Bellalta, C. Cano, A. Jonsson, and G. Neu, "Potential and pitfalls of multi-armed bandits for decentralized spatial reuse in WLANs," *Journal of Network and Computer Applications*, vol. 127, pp. 26 – 42, Feb. 2019.

[86] S. Pediaditaki, M. K. Marina, and D. Tyrode, "Traffic-aware channel width adaptation in long-distance 802.11 mesh networks," in *inProc. ACM MSWiM*, (Paphos, Cyprus), p. 261–270, Oct. 2012.

[87] J. B. Andersen, T. S. Rappaport, and S. Yoshida, "Propagation measurements and models for wireless communications channels," *IEEE Comms. Mag.*, vol. 33, no. 1, pp. 42–49, 1995.

[88] Y. Li, M. Sheng, C. Wang, X. Wang, Y. Shi, and J. Li, "Throughput–delay tradeoff in interference-free wireless networks with guaranteed energy efficiency," *IEEE Trans. Wireless Commun.*, vol. 14, pp. 1608–1621, Nov. 2015.

[89] B. Wang, Z. Ji, K. J. R. Liu, and T. C. Clancy, "Primary-prioritized markov approach for dynamic spectrum allocation," *IEEE Trans. Wireless Commun.*, vol. 8, no. 4, pp. 1854–1865, 2009.

[90] "802.11ac In-Depth." White Paper.

[91] V. Mathuranathan, "Log distance path loss or log normal shadowing model." https://www.gaussianwaves.com/2013/09/log-distance-path-loss-or-log-normal-shadowing-model/.

[92] A. Sepasi Zahmati, X. Fernando, and A. Grami, "Steady-state markov chain analysis for heterogeneous cognitive radio networks," in *IEEE Sarnoff Symposium*, (Princeton, USA), pp. 1–5, Apr. 2010.

[93] L. Zhang and K. Chin, "On devices selection in RF-energy harvesting wireless networks," *IEEE Syst. J.*, pp. 1–11, 2020.

Appendix A

Steady-State Probabilities for CTMC

The appendix shows how to derive the steady-state probabilities for CTMC in Chapter 5, which is given by

$$(\lambda_i + \lambda_j)\Pi_0 = \mu_i\Pi_1 + \mu_j\Pi_2 + \mu_i\Pi_3, \tag{A.1}$$

$$(\mu_i + \lambda_j)\Pi_1 = \lambda_i(1 - P_c)\Pi_0 + \mu_j\Pi_4, \tag{A.2}$$

$$(\mu_j + \lambda_i)\Pi_2 = \lambda_j\Pi_0 + \mu_i\Pi_4 + \mu_i\Pi_5, \tag{A.3}$$

$$(\mu_i + \lambda_j)\Pi_3 = \lambda_i P_c\Pi_0 + \mu_j\Pi_5, \tag{A.4}$$

$$(\mu_j + \mu_i)\Pi_4 = \lambda_j\Pi_1 + \lambda_i(1 - P_c)\Pi_2, \tag{A.5}$$

$$(\mu_i + \mu_j)\Pi_5 = \lambda_i P_c\Pi_2 + \lambda_j\Pi_3, \tag{A.6}$$

$$\sum_{k=0}^{5} \Pi_k = 1. \tag{A.7}$$

Note this is an over-determined system of linear equations. To solve it, we first extract (A.1)-(A.6), and write them into a new system of linear equations as follows

$$(\lambda_i + \lambda_j)X_0 = \mu_i X_1 + \mu_j X_2 + \mu_i X_3, \tag{A.8}$$

$$(\mu_i + \lambda_j)X_1 = \lambda_i(1 - P_c)X_0 + \mu_j X_4, \tag{A.9}$$

$$(\mu_j + \lambda_i)X_2 = \lambda_j X_0 + \mu_i X_4 + \mu_i X_5, \tag{A.10}$$

$$(\mu_i + \lambda_j)X_3 = \lambda_i P_c X_0 + \mu_j X_5, \tag{A.11}$$

$$(\mu_j + \mu_i)X_4 = \lambda_j X_1 + \lambda_i(1 - P_c)X_2, \tag{A.12}$$

$$(\mu_i + \mu_j)X_5 = \lambda_i P_c X_2 + \lambda_j X_3, \tag{A.13}$$

where X_k for $k \in \{0, 1, 2, 3, 4, 5\}$ are unknowns.

Now, re-arrange the above system of linear equations to its standard form as follows

$$-(\lambda_i + \lambda_j)X_0 + \mu_i X_1 + \mu_j X_2 + \mu_i X_3 = 0, \tag{A.14}$$

$$\lambda_i(1 - P_c)X_0 - (\mu_i + \lambda_j)X_1 + \mu_j X_4 = 0, \tag{A.15}$$

$$\lambda_j X_0 - (\mu_j + \lambda_i)X_2 + \mu_i X_4 + \mu_i X_5 = 0, \tag{A.16}$$

$$\lambda_i P_c X_0 - (\mu_i + \lambda_j)X_3 + \mu_j X_5 = 0, \tag{A.17}$$

$$\lambda_j X_1 + \lambda_i(1 - P_c)X_2 - (\mu_j + \mu_i)X_4 = 0, \tag{A.18}$$

$$\lambda_i P_c X_2 + \lambda_j X_3 - (\mu_i + \mu_j)X_5 = 0. \tag{A.19}$$

Next, write (A.14)-(A.19) into the following form: $AX = B$, where A is the matrix of coefficients, $X = [X_0, X_1, X_2, X_3, X_4, X_5]^T$ is the matrix of unknowns and $B = [0, 0, 0, 0, 0, 0]^T$ is the matrix of constants. Then, we perform elementary row operations to transform A into an upper triangular matrix, where all the entries below the main diagonal are zero. Let r_i represent the i-th row of a matrix.

$$A = \begin{bmatrix} -(\lambda_i+\lambda_j) & \mu_i & \mu_j & \mu_i & 0 & 0 \\ \lambda_i(1-P_c) & -(\mu_i+\lambda_j) & 0 & 0 & \mu_j & 0 \\ \lambda_j & 0 & -(\mu_j+\lambda_i) & 0 & \mu_i & \mu_i \\ \lambda_i P_c & 0 & 0 & -(\mu_i+\lambda_j) & 0 & \mu_j \\ 0 & \lambda_j & \lambda_i(1-P_c) & 0 & -(\mu_j+\mu_i) & 0 \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

We add r_2, r_3 and r_4 to r_1; add r_4 to r_2, so we have

$$\begin{bmatrix} 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ \lambda_j & 0 & -(\mu_j+\lambda_i) & 0 & \mu_i & \mu_i \\ \lambda_i P_c & 0 & 0 & -(\mu_i+\lambda_j) & 0 & \mu_j \\ 0 & \lambda_j & \lambda_i(1-P_c) & 0 & -(\mu_j+\mu_i) & 0 \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

We take row r_2 and multiply it by $(-\frac{\lambda_j}{\lambda_i})$ and add it to row r_3; take row r_2 and multiply it by $-P_c$ and add it to row r_4. This yields

$$\begin{bmatrix} 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & \frac{\lambda_j(\mu_i+\lambda_j)}{\lambda_i} & -(\mu_j+\lambda_i) & \frac{\lambda_j(\mu_i+\lambda_j)}{\lambda_i} & \mu_i-\frac{\lambda_j\mu_j}{\lambda_i} & \mu_i-\frac{\lambda_j\mu_j}{\lambda_i} \\ 0 & P_c(\mu_i+\lambda_j) & 0 & (\mu_i+\lambda_j)(P_c-1) & -P_c\mu_j & \mu_j(1-P_c) \\ 0 & \lambda_j & \lambda_i(1-P_c) & 0 & -(\mu_j+\mu_i) & 0 \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

Interchange row r_1 and row r_2, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & \frac{\lambda_j(\mu_i+\lambda_j)}{\lambda_i} & -(\mu_j+\lambda_i) & \frac{\lambda_j(\mu_i+\lambda_j)}{\lambda_i} & \mu_i-\frac{\lambda_j\mu_j}{\lambda_i} & \mu_i-\frac{\lambda_j\mu_j}{\lambda_i} \\ 0 & P_c(\mu_i+\lambda_j) & 0 & (\mu_i+\lambda_j)(P_c-1) & -P_c\mu_j & \mu_j(1-P_c) \\ 0 & \lambda_j & \lambda_i(1-P_c) & 0 & -(\mu_j+\mu_i) & 0 \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

We add r_2 to r_5; take r_4 and multiply by $-\frac{\lambda_j}{P_c\lambda_i}$ and add it to r_3, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & 0 & -(\mu_j+\lambda_i) & \frac{\lambda_j(\mu_i+\lambda_j)}{P_c\lambda_i} & \mu_i & \mu_i-\frac{\lambda_j\mu_j}{P_c\lambda_i} \\ 0 & P_c(\mu_i+\lambda_j) & 0 & (\mu_i+\lambda_j)(P_c-1) & -P_c\mu_j & \mu_j(1-P_c) \\ 0 & 0 & -\lambda_i P_c & -\lambda_j & 0 & \mu_i+\mu_j \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

We take r_2 and multiply by $\frac{P_c(\mu_i+\lambda_j)}{\lambda_j}$ and add it to r_4, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & 0 & -(\mu_j+\lambda_i) & \frac{\lambda_j(\mu_i+\lambda_j)}{P_c\lambda_i} & \mu_i & \mu_i-\frac{\lambda_j\mu_j}{P_c\lambda_i} \\ 0 & 0 & -\lambda_i P_c(1+\frac{\mu_i}{\lambda_j}) & -(\mu_i+\lambda_j) & P_c\mu_i(\frac{\mu_j+\mu_i}{\lambda_j}+1) & P_c\mu_i(\frac{\mu_j+\mu_i}{\lambda_j}+1)+\mu_j \\ 0 & 0 & -\lambda_i P_c & -\lambda_j & 0 & \mu_i+\mu_j \\ 0 & 0 & \lambda_i P_c & \lambda_j & 0 & -(\mu_i+\mu_j) \end{bmatrix}.$$

We add r_5 to r_6; take r_5 and multiply by $-(1+\frac{\mu_i}{\lambda_j})$ and add it to r_4, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & 0 & -(\mu_j+\lambda_i) & \frac{\lambda_j(\mu_i+\lambda_j)}{P_c\lambda_i} & \mu_i & \mu_i-\frac{\lambda_j\mu_j}{P_c\lambda_i} \\ 0 & 0 & 0 & 0 & P_c\mu_i(\frac{\mu_j+\mu_i}{\lambda_j}+1) & \mu_i(P_c-1)(\frac{\mu_i+\mu_j}{\lambda_j}+1) \\ 0 & 0 & -\lambda_iP_c & -\lambda_j & 0 & \mu_i+\mu_j \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We take r_5 and multiply by $-\frac{\mu_j+\lambda_i}{\lambda_iP_c}$ and add it to r_3, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & 0 & 0 & \frac{\lambda_j(\lambda_i+\lambda_j+\mu_i+\mu_j)}{P_c\lambda_i} & \mu_i & \mu_i-\frac{{\mu_j}^2+\lambda_i\mu_j+\mu_i\mu_j+\lambda_i\mu_i+\lambda_j\mu_j}{P_c\lambda_i} \\ 0 & 0 & 0 & 0 & P_c\mu_i(\frac{\mu_j+\mu_i}{\lambda_j}+1) & \mu_i(P_c-1)(\frac{\mu_i+\mu_j}{\lambda_j}+1) \\ 0 & 0 & -\lambda_iP_c & -\lambda_j & 0 & \mu_i+\mu_j \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We interchange r_3 and r_5; then interchange r_4 and r_5, so we have

$$\begin{bmatrix} \lambda_i & -(\mu_i+\lambda_j) & 0 & -(\mu_i+\lambda_j) & \mu_j & \mu_j \\ 0 & -\lambda_j & -\lambda_i & -\lambda_j & \mu_i+\mu_j & \mu_i+\mu_j \\ 0 & 0 & -\lambda_iP_c & -\lambda_j & 0 & \mu_i+\mu_j \\ 0 & 0 & 0 & \frac{\lambda_j(\lambda_i+\lambda_j+\mu_i+\mu_j)}{P_c\lambda_i} & \mu_i & \mu_i-\frac{{\mu_j}^2+\lambda_i\mu_j+\mu_i\mu_j+\lambda_i\mu_i+\lambda_j\mu_j}{P_c\lambda_i} \\ 0 & 0 & 0 & 0 & P_c\mu_i(\frac{\mu_j+\mu_i}{\lambda_j}+1) & \mu_i(P_c-1)(\frac{\mu_i+\mu_j}{\lambda_j}+1) \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= A.$$

Next, we use the back substitution method to solve for $X_5, X_4, \cdots, X_1$ one by one.

First, from the fifth row r_5 of A, we have

$$P_c\mu_i\left(\frac{\mu_j+\mu_i}{\lambda_j}+1\right)X_4+\mu_i(P_c-1)\left(\frac{\mu_i+\mu_j}{\lambda_j}+1\right)X_5=0. \tag{A.20}$$

We notice that X_5 and X_4 are *free variables*, which can take arbitrary values. Let $X_5 = P_c$, then

$$P_c\mu_i\left(\frac{\mu_j+\mu_i}{\lambda_j}+1\right)X_4+\mu_i(P_c-1)\left(\frac{\mu_i+\mu_j}{\lambda_j}+1\right)P_c=0. \tag{A.21}$$

We have $X_4 = 1 - P_c$.

From row r_4, we have

$$\frac{\lambda_j(\lambda_i+\lambda_j+\mu_i+\mu_j)}{P_c\lambda_i}X_3+\mu_iX_4+X_5\left(\mu_i-\frac{{\mu_j}^2+\lambda_i\mu_j+\mu_i\mu_j+\lambda_i\mu_i+\lambda_j\mu_j}{P_c\lambda_i}\right)=0. \tag{A.22}$$

Substitute $X_4 = 1 - P_c$ and $X_5 = P_c$ into (A.22),

$$\frac{\lambda_j(\lambda_i+\lambda_j+\mu_i+\mu_j)}{P_c\lambda_i}X_3+\mu_i(1-P_c)+P_c\left(\mu_i-\frac{{\mu_j}^2+\lambda_i\mu_j+\mu_i\mu_j+\lambda_i\mu_i+\lambda_j\mu_j}{P_c\lambda_i}\right)=0. \tag{A.23}$$

We have $X_3 = \frac{P_c\mu_j}{\lambda_j}$.

From row r_3, we have

$$-\lambda_iP_cX_2-\lambda_jX_3+(\mu_j+\mu_i)X_5=0. \tag{A.24}$$

Substitute $X_3 = \frac{P_c\mu_j}{\lambda_j}$ and $X_5 = P_c$ into (A.24),

$$-\lambda_iP_cX_2-\lambda_j\frac{P_c\mu_j}{\lambda_j}+(\mu_j+\mu_i)P_c=0. \tag{A.25}$$

We have $X_2 = \frac{\mu_i}{\lambda_i}$.

From row r_2, we have

$$-\lambda_jX_1-\lambda_iX_2+\lambda_jX_3+(\mu_j+\mu_i)X_4+(\mu_j+\mu_i)X_5=0. \tag{A.26}$$

Substitute $X_2 = \frac{\mu_i}{\lambda_i}$, $X_3 = \frac{P_c\mu_j}{\lambda_j}$, $X_4 = 1 - P_c$ and $X_5 = P_c$ into (A.26),

$$-\lambda_j X_1 - \lambda_i \frac{\mu_i}{\lambda_i} + \lambda_j \frac{P_c\mu_j}{\lambda_j} + (\mu_j + \mu_i)(1 - P_c) + (\mu_j + \mu_i)P_c = 0. \tag{A.27}$$

We have $X_1 = \frac{(1-P_c)\mu_j}{\lambda_j}$.

From row r_1, we have

$$\lambda_i X_0 - (\mu_i + \lambda_j)X_1 - (\mu_i + \lambda_j)X_3 + \mu_j X_4 + \mu_j X_5. \tag{A.28}$$

Substitute $X_1 = \frac{(1-P_c)\mu_j}{\lambda_j}$, $X_2 = \frac{\mu_i}{\lambda_i}$, $X_3 = \frac{P_c\mu_j}{\lambda_j}$, $X_4 = 1 - P_c$ and $X_5 = P_c$ into (A.28),

$$\lambda_i X_0 - \frac{(\mu_i + \lambda_j)(1 - P_c)\mu_j}{\lambda_j} - \frac{(\mu_i + \lambda_j)P_c\mu_j}{\lambda_j} + \mu_j(1 - P_c) + \mu_j P_c = 0. \tag{A.29}$$

We have $X_0 = \frac{\mu_i\mu_j}{\lambda_i\lambda_j}$.

Next, we need find the *particular* solution that satisfies $\sum_{k=0}^{5} \Pi_k = 1$. First, we calculate

$$\begin{aligned} X_0 + X_1 + X_2 + X_3 + X_4 + X_5 &= \frac{\mu_i\mu_j}{\lambda_i\lambda_j} + \frac{(1 - P_c)\mu_j}{\lambda_j} + \frac{\mu_i}{\lambda_i} + \frac{P_c\mu_j}{\lambda_j} + (1 - P_c) + P_c \\ &= \frac{\lambda_i\mu_j + \lambda_i\lambda_j + \mu_i\mu_j + \lambda_j\mu_i}{\lambda_i\lambda_j} \end{aligned} \tag{A.30}$$

As $\sum_{k=0}^{5} \Pi_k = 1$, we have

$$\begin{aligned} &\Pi_0 + \Pi_1 + \Pi_2 + \Pi_3 + \Pi_4 + \Pi_5 = \frac{\lambda_i\mu_j + \lambda_i\lambda_j + \mu_i\mu_j + \lambda_j\mu_i}{\lambda_i\lambda_j} \cdot \frac{\lambda_i\lambda_j}{\lambda_i\mu_j + \lambda_i\lambda_j + \mu_i\mu_j + \lambda_j\mu_i} \\ &= (X_0 + X_1 + X_2 + X_3 + X_4 + X_5) \cdot \frac{\lambda_i\lambda_j}{\lambda_i\mu_j + \lambda_i\lambda_j + \mu_i\mu_j + \lambda_j\mu_i} \\ &= 1 \end{aligned} \tag{A.31}$$

Therefore, this means we can obtain Π_k from

$$\Pi_k = \frac{X_k}{\frac{\lambda_i\mu_j+\lambda_i\lambda_j+\mu_i\mu_j+\lambda_j\mu_i}{\lambda_i\lambda_j}}, \quad k \in \{0, 1, \cdots, 5\} \tag{A.32}$$

Then, we have the solution for (A.1)-(A.7) as follows

$$\Pi_0 = \Gamma\mu_i\mu_j, \tag{A.33}$$

$$\Pi_1 = \Gamma(1 - P_c)\lambda_i\mu_j, \tag{A.34}$$

$$\Pi_2 = \Gamma\lambda_j\mu_i, \tag{A.35}$$

$$\Pi_3 = \Gamma P_c\lambda_i\mu_j, \tag{A.36}$$

$$\Pi_4 = \Gamma(1 - P_c)\lambda_i\lambda_j, \tag{A.37}$$

$$\Pi_5 = \Gamma P_c\lambda_i\lambda_j, \tag{A.38}$$

where

$$\Gamma = (\lambda_i\lambda_j + \lambda_i\mu_j + \lambda_j\mu_i + \mu_i\mu_j)^{-1}. \tag{A.39}$$